VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webové rozhraní IDS systému SNORT

Web Interface for Snort IDS

2019 Bc. Markéta Glatzová

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Markéta Glatzová**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 1801T064 Informační a komunikační bezpečnost

Téma: Webové rozhraní IDS systému Snort
Web Interface for Snort IDS

Jazyk vypracování: čeština

Zásady pro vypracování:

Snort je síťový Intrusion Detection System (IDS) založený na pravidlech. Na základě odposlechu síťové komunikace hledá známé vzorky útoků a v případě nalezení je schopen provádět další akce. Kromě generování protokolu události je schopen zaznamenávat paketová data na disk, kde mohou sloužit k další analýze. Cílem diplomové práce je vytvořit nástavbu aplikace Snort umožňující přehlednou analýzu logů skrz webové rozhraní. Inspirací mohou být projekty BASE, OSSIM nebo SnortALog. Vyvinuté rozhraní by mělo být schopné provozu i na méně výkonných routerech typu Alix.

1. Seznamte se s IDS systémy, jejich funkcemi a konfigurací.
2. Proveďte rešerši grafických a webových nástaveb IDS systému Snort.
3. Navrhněte a implementujte vlastní rozhraní umožňující přehlednou vizualizaci zachycených útoků s možností stáhnout zachycené paketové data.
4. Vytvořte vhodnou statistiku popisující charakter a typ útoků.
5. Otestujte navržené řešení a výstupy porovnejte s alespoň dvěma jinými systémy umožňujícími vizualizaci protokolů systému Snort.
6. Shrňte dosažené výsledky a popište možnosti dalšího vývoje.

Seznam doporučené odborné literatury:

[1] Chris McNab, Network Security Assessment: Know Your Network, O'Reilly Media; 3 edition, 2016, ISBN 978-1491910955
[2] Rafeeq Rehman, Intrusion Detection With SNORT, Apache, MySQL, PHP, And ACID, Pearson Technology Group, 2007, ISBN 978-0131407336
[3] Diyar Salah Fadhil, Samir Al-Khayatt, Automatic Intrusion Prevention Technique to Improve Network Security, LAP LAMBERT Academic Publishing, 2016, ISBN 978-3659962011

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě 15. dubna 2019

.....

Ráda bych na tomto místě poděkovala svému vedoucímu diplomové práce panu Mgr. Ing. Michalovi Krumniklovi, Ph.D. za jeho vstřícný přístup a ochotu, se kterou mi pomáhal při tvorbě této práce.

Abstrakt

Cílem této diplomové práce je vytvořit webové rozhraní pro IDS systém Snort, který se používá pro detekci útoků. V první části této práce jsou objasněny pojmy spojené s IDS systémy, jejich funkce a porovnání vybraných IDS systémů. Tyto teoretické poznatky jsou dále uplatněny v části analytické a praktické. Analytická část obsahuje porovnání základních charakteristik starších webových rozhraní a také porovnání jejich funkcí s webovým rozhraním vytvořeným v části praktické. V praktické části je zpracováno samotné webové rozhraní IDS systému Snort. V závěru práce jsou shrnuty dosažené výsledky a popsány možnosti dalšího vývoje.

Klíčová slova: IDS, IPS, NIDS, HIDS, Snort, Ruby, Ruby on Rails, webové rozhraní

Abstract

The goal of this master thesis is to create a web-based interface for the IDS system Snort, used for network intrusion detection prevention. The first part of the thesis describes the basic terms of the various IDS systems, their function and feature comparison of selected products. All of the theoretical research is further used in the analytical and implementation part of the thesis. Analytical part includes a comparison of fundamental characteristics of older web-based interfaces and comparison with the interface created in the practical part of the thesis. The implementation is focused on the actual creation of the web interface and It's deployment with Snort IDS. The thesis concludes with an evaluation of the results and a description of another possible extensions.

Key Words: IDS, IPS, NIDS, HIDS, Snort, Ruby, Ruby on Rails, web interface

Obsah

Seznam použitých zkratek a symbolů

IDS	–	Intrusion Detection System
IPS	–	Intrusion Prevention System
NIDS	–	Network Intrusion Detection System
HIDS	–	Host Intrusion Detection System
RoR	–	Ruby on Rails
HTML	–	Hypertext Markup Language
Haml	–	HTML abstraction markup language
CSS	–	Cascading Style Sheets
AJAX	–	Asynchronous JavaScript and XML

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V dnešním světě plném vyspělých technologií a internetu je bezpečnost velice důležitá. Zajištění bezpečnosti je důležité především v rámci podnikových sítí, ale i v sítích domácích. V domácích sítích si většinou vystačíme s obyčejným firewallem, ale v podnikové síti již nemusí stačit. Je více, než vhodné zkombinovat firewall s IDS. V ideálním případě pak využít kombinaci IDS/IPS s firewallem. Firewall samotný funguje jakoby pomyslná „zeď“, která nepropustí do sítě žádný škodlivý, nebo nebezpečný provoz. Bohužel, ale síla této pomyslné „zdi“ závisí na hodně aspektech a ne vždy funguje správně. V případě, že máme navíc správně nakonfigurovaný IDS systém, tak dostaneme hlášení o veškerém nebezpečném, nebo podezřelém provozu, který přes firewall prošel. Jak tyto výstupy přesně vypadají záleží pak na zvoleném IDS systému. Pokud se rozhodneme pro kombinaci IDS/IPS, tak IPS zvládá nejen provoz sledovat a oznamovat nám podezřelé aktivity, ale také podnikat samotná opatření k zamezení vniknutí škodlivého provozu do podnikové sítě, nebo jiné sítě.

Jako téma jsem si vybrala webové rozhraní IDS systému Snort, protože mne Snort ze všech IDS/IPS oslovil nejvíce. Je volně dostupný a hojně využívaný, ale zároveň jsem k němu nenalezla žádné moderní grafické rozhraní. Proto jsem se rozhodla, že se pokusím jedno takové moderní rozhraní ke Snortu vytvořit. Práce je členěna do čtyř hlavních částí. První část se zabývá praktickými znalostmi týkajících se IDS, IPS a Snortu, a porovnává různé druhy IDS i jejich výstupy. V části druhé se zabývám rešerší grafických a webových nástaveb IDS systému Snort. Třetí část je již praktická a zabývá se samotným webovým rozhraním vytvořeným v rámci této diplomové práce. Poslední část je věnována testování navrženého řešení, porovnání výstupů s dalšími systémy umožňujícími vizualizaci protokolů systému Snort. V samotném závěru pak shrnuji dosažené výsledky a zmiňuji možnosti dalšího vývoje.

Součástí práce jsou zdrojové kódy webového rozhraní vytvořeného v praktické části práce.

2 Systémy IDS

Intrusion detection system (IDS), neboli také systém detekce narušení je systém, který monitoruje síťovou komunikaci, monitoruje podezřelou aktivitu a upozorňuje správce systému, nebo sítě.

Zjišťování a hlášení anomálií je primární funkcí IDS. Dále jsou ale také některé systémy detekce narušení schopny, v případě zjištění škodlivé činnosti, nebo také anomálního provozu, provádět další akce pro zajištění bezpečného provozu.

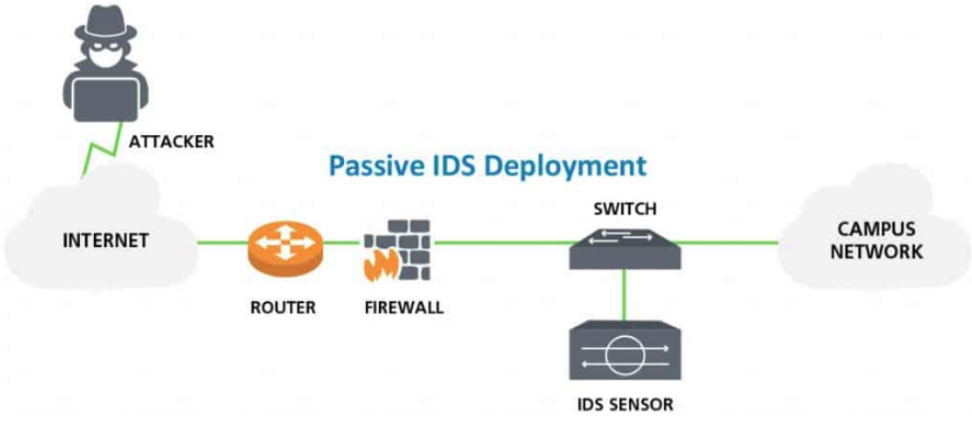
Z tohoto hlediska můžeme systémy detekce narušení rozdělit na aktivní a pasivní. Pasivní systémy jednoduše jen detekují a upozorňují na podezřelé aktivity. V případě zjištění podezřelé, nebo škodlivé aktivity se provádí zápis do výstupního souboru, je vygenerováno upozornění a následně odesláno správci, nebo uživateli a dále je pouze na nich, aby podnikli správné kroky k blokování dané aktivity, nebo podnikli jiná opatření.

Kromě výše zmíněného, ale také může IDS přijmout předdefinované proaktivní akce, aby reagoval na hrozbu. Obvykle to znamená blokování jakéhokoli dalšího síťového provozu ze zdrojové IP adresy, nebo IP adresy uživatele.

Přestože IDS monitoruje síť pro potenciálně škodlivé činnosti, je také náchylný k falešným poplachům. Je proto třeba vyladit IDS již při jeho první implementaci. To znamená správně ho nakonfigurovat tak, aby měl přehled, jak vypadá normální provoz v síti, ve srovnání s potenciálně škodlivým provozem.

2.1 Rozdělení IDS podle jeho umístění v síti

IDS může fungovat různými způsoby a detekuje podezřelé činnosti pomocí různých metod. IDS lze rozdělit do následujících tří skupin podle jeho umístění v síti. Umístění pasivního IDS v síti je vyobrazeno na obrázku 1.



Obrázek 1: Umístění pasivního IDS v síti, obrázek převzat z: [18]

2.1.1 Síťově orientovaný IDS (NIDS)

Síťový systém detekce narušení sítě (NIDS) se používá k monitorování a analýze síťového provozu za účelem ochrany systému před hrozbami založenými na síti. NIDS je umístěn ve strategickém bodě, nebo bodech v síti a monitoruje provoz na všech zařízeních v síti. NIDS čte všechny příchozí pakety a vyhledává všechny podezřelé vzory. V případě objevení hrozby se může systém na základě její závažnosti rozhodnout k podniknutí dalších kroků, kterými mohou být například rozesílání oznámení správcům, nebo také nejčastější blokování zdrojové IP adresy z přístupu k síti. Mezi NIDS patří například Cisco Secure IDS (dříve NetRanger), Hogwash, Dragon, E-Trust IDS, ale také Snort.

2.1.2 Hostitelsky orientovaný IDS (HIDS)

Systém detekce narušení hostitele (HIDS) je uzlově orientovaný IDS, který monitoruje počítačový systém, na kterém je nainstalován, aby detekoval vniknutí, nebo zneužití. HIDS je spuštěn na jednotlivých počítačích, nebo zařízeních v síti. HIDS monitoruje pouze příchozí a odchozí pakety a v případě odhalení nežádoucí činnosti tuto činnost nejen zaznamená, ale také vytváří oznámení. Mezi HIDS patří Dragon Squire, Emerald eXpert-BSM, NFR HID, Intruder Alert.

2.1.3 Hybridní IDS

Hybridní IDS je kombinací NIDS a HIDS. Sleduje tedy jak aktivity přímo na hostitelském systému, tak síťový provoz. Příkladem hybridního IDS je například Sagan, Security Onion a SolarWinds Log and Event Manage.

2.2 Způsoby detekce IDS

IDS může detekovat podezřelé činnosti pomocí různých metod. Nebezpečné aktivity mohou být v síti nalezeny především pomocí porovnávání charakteristik s charakteristikami již známých útoků, dlouhodobým sledováním provozu a následném rozpoznání anomálií. Existují ale i další metody. Nejpoužívanější bývají systémy hybridní, které zvládají kombinovat více způsobů detekce.

2.2.1 IDS zaměřený na charakteristiky

IDS zaměřený na charakteristiky monitoruje pakety v síti a porovnává je s databází vzorů útoků, neboli charakteristik, známých jako škodlivé hrozby. Je to podobné způsobu, který používá většina antivirových programů k detekci malwaru. Problém spočívá v tom, že mezi novou hrozbou, která se objeví a charakteristikou pro zjištění hrozby, bude zpoždění. Během tohoto zpoždění není IDS schopen detekovat novou hrozbu.

2.2.2 IDS zaměřený na statistické anomálie

IDS, který je založen na statistických anomáliích, sleduje síťový provoz a porovnává ho s provozem standardním. Výchozí hodnota určuje, co je pro danou síť „normální“. Výchozí hodnotou může být například obecně používaná šířka pásma, nebo běžně využívané porty a jejich koncová zařízení. Pokud je provoz detekován jako anomální, nebo se výrazně liší od standardního provozu, tak upozorní administrátora, nebo uživatele.

2.2.3 Ostatní způsoby detekce

Mezi další způsoby můžeme řadit systémy korelační. Tyto systémy vyhledávají souvislosti mezi jevy probíhajícími na více místech. Korelační systémy se vyznačují svou složitostí nasazení v systému, ale i menší pravděpodobností vzniku falešného poplachu. Příkladem může být například korelace mezi Nessusem a Snortem. Snort detekuje útok na přetečení zásobníku (stack overlow) a Nessus jakožto bezpečnostní scanner objevil aplikaci zranitelnou tímto druhem útoku, výsledkem je pak vygenerování varování s vysokou prioritou.

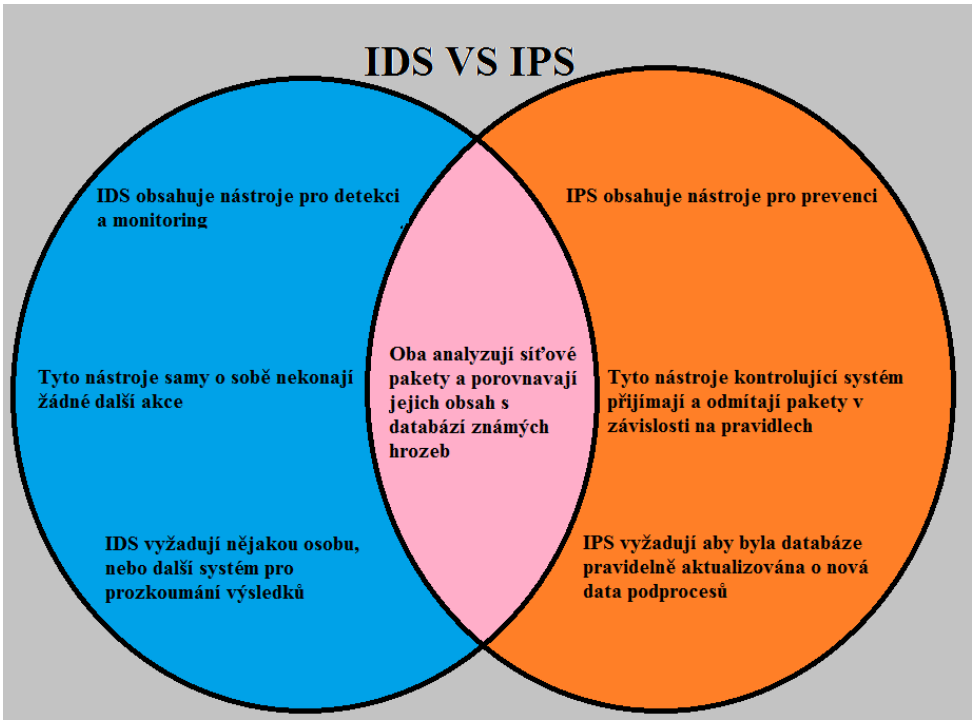
2.3 IPS systémy a rozdíly oproti IDS

IDS a IPS jsou dost odlišné systémy, ale technologie, které používají k detekci/prevenci bezpečnostních problémů jsou velmi podobné. Společný mají ale pouze základ a v síti jsou umístěni na různých místech, mají různé funkce a řeší různé problémy. IPS lze nejlépe srovnat s firewallem. Typický firewall v podnikové síti obsahuje mnoho pravidel, jedná se o řády stovek až tisíců. Brána firewall dostane paket a začne v seznamu pravidel hledat pravidlo, které říká, že se daný paket může povolit. Pokud žádné takové pravidlo nenajde, paket neprojde dál do sítě, jelikož existuje konečné pravidlo, které odpírá veškerý provoz, pro který není nalezeno pravidlo, které by paket povolilo. IPS funguje v podstatě stejně, ale naopak. IPS hledá mezi pravidly pravidlo, díky kterému by mohl paket odmítnout a pokud ho nenalezne, propustí ho. Konečné pravidlo tedy u IPS propustí veškerý provoz, který neshledá podezřelým. Hlavním důvodem využití IPS je jeho schopnost blokovat známé útoky napříč sítí. Je to vynikající způsob, jak rychle blokovat známé útoky, zejména ty, které používají běžné, nebo dobře známé nástroje používané ke zneužití. Na Obrázku 2 je vyobrazeno zapojení aktivního IPS v síti.

Obrázek 2: Umístění aktivního IPS v síti, obrázek převzat z: [18]

IPS fungují jako rozšíření IDS. Kromě samotných detekcí hrozeb byly v IPS přidány také schopnosti tyto hrozby blokovat. IPS lze považovat za jakoby „strážce“, který je aktivně přítomen v síti, má zabránit příchozím útokům a zastavit již probíhající útoky. Tento „strážce“ tedy nemusí být vždy úspěšný v zabránění proniknutí tzv.: „vetřelce“, neboli narušitele do sítě, ale v případě, že narušitel vnikne do sítě i přes veškerá nastavená pravidla, dokáže zabránit dalším škodám.

Kombinace IDS a IPS se stala nejčastější formou obrany proti útokům. Obrázek 1 shrnuje rozdíly mezi IDS a IPS, ale lze na něm pozorovat i vlastnosti společné.



Obrázek 3: IDS systémy versus IPS systémy

3 Porovnání a popis nejčastěji využívaných IDS systémů

IDS systémů existuje celá řada. Některé jsou volně dostupné, ale za některé si musíme i zaplatit. Nejvyužívanějšímu IDS systému Snortu je vyhrazena celé kapitola 4, tato kapitola se věnuje pouze shrnutí ostatních IDS systémů. Pro bližší srovnání jsem si vybrala následující hojně využívané a volně dostupné IDS systémy [3] .

3.1 Surikata

Na rozdíl od jiných systémů IDS / IPS, má Surikata ke Snortu nejblíže. Klíčovou výhodou Surikaty je, že shromažďuje data na aplikační vrstvě. Surikata čeká, až se data v paketech sestaví ještě předtím, než se provádí analýza. Rozezná tedy i škodlivé vzory rozdělené přes více TCP paketů. Ačkoliv systém pracuje na aplikační vrstvě, je schopen monitorovat aktivitu protokolu na nižších úrovních, jako jsou IP, TLS, ICMP, TCP a UDP. Zkoumá provoz v reálném čase, monitoring se nezabývá pouze strukturou paketů, ale může zkoumat i certifikáty TLS a zaměřit se na požadavky HTTP a DNS. Surikata je se Snortem kompatibilní a lze použít i stejná pravidla VRT. Nástroje třetích stran, jako například Base, Snorby, Squil a Anaval, které lze integrovat se Snortem, mohou být připojeny také k Surikatě. Přístup Snort komunity k udělování tipů, poskytnutí bezplatných pravidel a pod. může být velkým přínosem i pro uživatelé Surikaty. Vestavěný skriptovací modul umožňuje kombinovat pravidla a získat přesnější profil detekce, než Snort. Surikata má podobnou architekturu jako Snort, který se spoléhá na detekci vzorů a anomálií. Surikata má inteligentní architekturu zpracování, která umožňuje hardwarovou akceleraci pomocí mnoha různých procesorů pro simulaci vícevláknové aktivity. Částečně může vše běžet i na naší grafické kartě. Díky tomuto rozdělení úkolů se udržuje zátěž což je výhodou, jelikož problém NIDS spočívá v tom, že je velmi těžký na zpracování. Suricata má velmi úhledný přístrojový panel, který obsahuje grafiku, která usnadňuje analýzu a rozpoznávání problémů. Surikata je k dispozici zdarma. Surikata nenabízí jen síťově orientované IDS a IPS, ale má i další funkce:

- Monitorování zabezpečení sítě (Network Security Monitoring – NSM)
- Offline analýza PCAP souborů
- Nahrávání provozu pomocí pcap loggeru
- Unixový soketový režim pro automatizované zpracování PCAP souborů
- Pokročilá integrace s firewallem Netfilter

Podporovanými operačními systémy jsou: Linux, Mac OS, Windows, FreeBSD, OpenBSD, Fedora, CentOS
Poslední stabilní verze: 4.1.3

3.1.1 Výstupy Surikaty

Surikata podporuje různé formy výstupu[6], které jsou blíže popsány v paragrafech uvedených níže.

3.1.1.1 Jednořádkový výstup pro výstrahy (fast.log) Tento výstup obsahuje výstrahy skládající se z jednoho řádku. Níže je ukázka jednoho takového řádku souboru fast.log. Každý řádek začíná informací o čase zachycení události takzvaného „timestampu“. Dále řádek obsahuje například klasifikaci útoku, jeho prioritu, protokol, zdrojovou a cílovou adresu.

```
10/03/19-12:11:59.667372 [**] [1:2009187:4] ET WEB_CLIENT ACTIVEX iDefense
COMRaider ActiveX Control Arbitrary File Deletion [**] [Classification: Web
Application Attack] [Priority: 3] {TCP} xx.xx.232.144:80 -> 192.168.1.4:56068
```

3.1.1.2 Formát Eve (Extensible Event Format) Jedná se o výstup ve formátu JSON obsahující výstrahy a události. Umožňuje snadnou integraci s nástroji třetích stran, jako je například logstash. EVE JSON struktura může vypadat následovně.

```
{
       "timestamp": "2009-11-24T21:27:09.534255",
       "event_type": "alert",
       "src_ip": "192.168.2.7",
       "src_port": 1041,
       "dest_ip": "x.x.250.50",
       "dest_port": 80,
       "proto": "TCP",
       "alert": {
          "action": "allowed",
          "gid": 1,
          "signature_id" :2001999,
          "rev": 9,
          "signature": "ET MALWARE BTGrab.com Spyware Downloading Ads",
          "category": "A Network Trojan was detected",
          "severity": 1
       }
   }
```

3.1.1.3 Unified2 alert výstup pro použití s Barnyard2 (unified2.alert)

Tento formát výstupu má binární podobu a je kompatibilní s Unified2 výstupem dalších populárních IDS. Výstup podporuje události IPv4 i IPv6. Unified2 soubor lze využít v kombinaci s Barnyardem2 a dalšími nástroji, které pracují s tímto formátem. Barnyard2 je nástroj s jehož pomocí jsou data přeneseny do databáze MySQL, nebo Postgresql.

3.1.1.4 Jednořádkový výstup HTTP žádostí (http.log)

Tento výstup uchovává informace o všech událostech HTTP. Obsahuje HTTP, hostname, URI a User-Agent a tyto informace ukládá v souboru http.log. Tento typ logování může být realizován i pomocí funkce Eve-log. Řádek v souboru http.log může vypadat například následovně.

```
07/01/2014-04:20:14.338309 vg.no [**] / [**] Mozilla/5.0 (Macintosh; Intel Mac
OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.114
    Safari/
537.36 [**] 192.168.1.6:64685 -> 195.88.54.16:80
```

3.1.1.5 Jednořádkový výstup DNS dotazů a odpovědí (dns.log)

Výstup uchovává informace o všech DNS událostech, dotazech i odpovědích. Obsahuje typ činnosti DNS, která byla provedena, požadována/zodpovězena, název domény a relevantní data, jako například data klienta, serveru, ttl a zdroje záznamu. Stejně jako u předchozího typu výstupu lze logování provést pomocí funkce Eve-log, která nabízí snadnější analýzu. Příklad záznamu DNS dotazu s předchozí odpovědí:

```
07/01/2014-04:07:08.768100 [**] Query TX 14bf [**] zeustracker.abuse.ch [**]
A [**] 192.168.1.6:37681 -> 192.168.1.1:53
07/01/2014-04:07:08.768100 [**] Response TX 14bf [**] zeustracker.abuse.ch
    [**]
A [**] TTL 60 [**] 205.188.95.206 [**] 192.168.1.1:53 -> 192.168.1.6:37681
```

3.1.1.6 Paket log (pcap-log)

Pomocí volby pcap-log můžeme ukládat všechny pakety, které Surikata zaregistrovala do jednoho výstupního souboru s názvem _log.pcap_. Defaultně se tento soubor vytváří v souboru default-log-dir, ale lze ho vytvořit kdekoliv jinde, a to změnou absolutní cesty v souboru „yaml“. Soubor lze otevřít jakýmkoliv programem, který podporuje formát souboru pcap, jako například: Wireshark, TCPdump, Suricata, Snort a mnoho dalších. Vytváření tohoto typu výstupu lze buď povolit, nebo zakázat a existuje limit velikosti souboru, který je 32MB. Po dosažení maximální velikosti se vytvoří soubor nový. Ve výchozím nastavení jsou zaznamenány všechny pakety s výjimkou datových toků TCP mimo stream.reassembly.depth a šifrovaných proudů po výměně klíčů.

3.1.1.7 Podrobný záznam výstrah (alert-debug.log)

Tento výstup obsahuje doplňující informace o výstrahách (alertech).

3.1.1.8 Statistiky

Ve statistikách můžete nastavit možnosti pro stats.log. Když povolíme možnost stats.log, tak můžeme nastavit dobu v sekundách, po kterou chceme výstupní data do výstupního souboru zapisovat.

3.1.1.9 Syslog

Touto volbou je možné posílat všechny, výstrahy a událostí do syslogu.

3.1.1.10 Jednořádkové informace o zahozených paketech

Pokud Surikata pracuje v režimu IPS, může na základě pravidel vypnout pakety. Vynechané pakety se uloží do souboru drop.log, formátu protokolu Netfilter.

3.1.1.11 Ukládání souborů (File Extraction)

Ukládání souborů file-store umožňuje ukládání extrahovaných souborů na disk a nastavit místo, kde jsou tyto soubory uloženy.

3.2 OSSEC

OSSEC je open source, neboli volně dostupný HIDS, který nabízí několik dalších modulů, které mohou být použity se základními funkcemi samotného IDS. Kombinuje všechny aspekty HIDS, monitorování logů a Security Incident Management (SIM) / Security information and Event Management (SIEM) společně v jednoduchém řešení. Kromě detekce narušení může OSSEC provádět monitorování integrity souborů a detekci rootkitů pomocí výstrah v reálném čase, které jsou centrálně řízeny a se schopností vytvářet různé politiky, v závislosti na potřebách sítě. OSSEC má různé klíčové benefity, kterými například jsou:

- Požadavky na shodu – OSSEC pomáhá zákazníkům splnit specifické požadavky na shodu, jako například PCI a HIPPA. Toto umožňuje zákazníkům zjistit a upozornit je na neoprávněné úpravy systémů souborů a škodlivého chování vloženého do výstupních souborů komerčních produktů, stejně jako u vlastních aplikací.
- Multiplatformní – OSSEC může běžet na různých platformách.
- Varování v reálném čase a konfigurovatelné výstrahy – OSSEC umožňuje zákazníkům konfigurovat incidenty, na které chtějí být upozornění a umožňuje jim zaměřit se na zvyšování priority kritických incidentů, nad běžným šumem v jakémkoliv systému. Integrace spolu se smtp, sms a syslogem umožňují odesílání upozornění na e-mailovou adresu zákazníka.
- Integrace se stávající infrastrukturou – OSSEC je integrován se současnými produkty od zákazníků, jako jsou SIM / SEM (Security Incident Management / Security Events Management) produkty pro centralizované reportování a korelaci událostí.

- Centralizované řízení – OSSEC poskytuje jeden zjednodušený centralizovaný server pro správu politik ve více operačních systémech.

Podporovanými operačními systémy jsou: Linux, Mac OS, Windows, FreeBSD, OpenBSD, Fedora, CentOS, Solaris
Poslední stabilní verze: 4.1.3

3.3 Zeek

Zeek byl dříve známý jako Bro. Zeek je téměř jako Security Onion, používá však více, než jen pravidla, aby zjistil, odkud útoky přicházejí. Zeek používá kombinaci nástrojů. Zeek je považován za NIDS založený na specifikaci. Využívá širokou škálu modulů pro analýzu protokolů, kontrolu provozu a rozhodování o shodě s různými normami. Zeek se dá použít jako silný doplněk ke Snortu.
Zeek nabízí mnoho funkcí, kterými jsou například:

- Flexibilní síťový bezpečnostní monitoring sítě s korelací událostí.
- Kontrola veškerého provozu.
- Detekce útoků.
- Vytvoření log záznamu.
- Distribuovaná analýza.
- Plná programovatelnost.

Podporovanými operačními systémy jsou: Linux, Mac OS, FreeBSD, OpenBSD, Fedora, CentOS, Solaris
Poslední stabilní verze: 2.6.1

3.4 Security Onion

Security Onion se používá nejen pro monitorování sítě, ale i k detekci narušení. Jedná se o distribuci Linuxu založenou na Ubuntu. Může monitorovat více VLAN, nebo podsítí a funguje dobře ve VMware a dalších virtuálních prostředích. Security Onion lze použít jako IDS, ale neexistuje momentálně podpora i pro IPS. Onion spolupracuje s mnoha dalšími systémy pro detekci narušení, jakými například jsou: Elasticsearch, Logstash, Kibana, Snort, Suricata, Bro, Wazuh, Sguil, Squert, CyberChef, NetworkMiner a mnoho dalších bezpečnostních nástrojů.
Security onion podporuje více typů dat. Těmito typy jsou například:

- Data výstrah – HIDS upozornění z Wazuhu a NIDS upozornění ze Snortu, nebo Surikaty.
- Data assetů – Data aktiv z Bro.

- Data s úplným obsahem – Plně zachycené pakety z netsniff-ng.
- Data hosta – Hostitelská data získaná z Beatsu, Wazuhu, syslogu a dalších.
- Session data – Data relace z Bro.
- Transakční data – Http, ftp, dns, ssl a další typy logů z Bro.

Podporovanými operačními systémy jsou: Linux
Poslední stabilní verze: 16.04.5.6_20190110

3.5 OpenWIPS-NG

OpenWIPS-NG je bezdrátový IDS / IPS, který se spoléhá na server, senzory a rozhraní. Jedná se o opensource. OpenWIPS-NG je modulární a umožňuje administrátorovi stahovat plug-iny pro další funkce.
WIPS-ng se skládá ze tří částí. Těmito částmi jsou:

- Senzory – Zařízení které zachytí bezdrátový provoz a odešle jej na server k následné analýze. Reaguje také na útoky.
- Server – Agreguje data ze všech senzorů, provádí jejich analýzu a reaguje na útoky. Vytváří také upozornění na útoky.
- Rozhraní – GUI neboli uživatelské rozhraní spravuje server a zobrazuje informace o hrozbách v dané bezdrátové síti.

Podporovanými operačními systémy jsou: Linux
Poslední stabilní verze: 0.1 beta 1

	HIDS/NIDS	Linux	Windows	Mac OS	FreeBSD	OpenBSD	Solaris	Fedora	CentOS	Poslední verze
Snort	NIDS	✓	✓		✓			✓	✓	2.9.12
Surikata	NIDS	✓	✓	✓	✓	✓		✓	✓	4.1.3
OSSEC	HIDS	✓	✓	✓	✓	✓	✓	✓	✓	3.2.0
Zeek	NIDS	✓		✓	✓	✓	✓	✓	✓	2.6.1
Security Onion	HIDS/NIDS	✓								16.04.5.6_20190110
Open WIPS-NG	NIDS	✓								0.1 beta 1

Tabulka 1: Srovnání IDS systémů

4 IDS Snort

Snort je jednovláknový systém pro detekci narušení (IDS) a pro prevenci průniku (IPS), který byl původně vyvinut Martinem Roeschem v roce 1998. Jedná se o open source systém, neboli volně dostupný zdarma pod licencí GPL (General Public Licence). Je schopen analýzy v reálném čase a k logování paketů. Funguje pod operačním systémem Windows a GNU/Linux. Díky jeho stabilitě a dobré detekci škodlivých paketů je Snort velice často využívaným nástrojem pro zajištění bezpečnosti.

Snort je založen na libpcap, což je nástroj pro zachycení paketů, který je široce používán v TCP/IP analyzátorech. Prostřednictvím analýzy protokolů a porovnávání obsahu paketů detekuje Snort metody útoků, včetně odmítnutí služby (DOS), přetečení vyrovnávací paměti (overflow), útoku CGI, skenování tajných portů a sond SMB. Při zjištění podezřelého chování odešle Snort výstrahu v reálném čase do syslogu, do samostatného souboru výstrah, nebo se zobrazí ve vyskakovacím okně.
Podporovanými operačními systémy jsou: Linux, Windows, FreeBSD, Fedora, CentOS
Poslední stabilní verze: 2.9.12

4.1 Režimy Snortu

Snort dokáže pracovat ve třech různých režimech. Režimy konfigurace jsou následující:

1. Režim sniffer (Sniffer mode)
2. Režim logování paketů (Packet logger mode)
3. Režim detekce narušení sítě (Network intrusion detection mode)

4.1.1 Režim sniffer (Sniffer mode)

Režim Sniffer, zobrazuje pakety, které procházejí po síti. Může být nastaven tak, aby zobrazoval různé typy paketů (TCP, UDP, ICMP), jakož i to, co samotné pakety obsahují, a to buď záhlaví, nebo paketová data.
Pro zobrazení IP a TCP/UDP/ICMP hlaviček použijeme příkaz: **snort -v**
Zachycený paket, který je vyobrazen na obrazovce vypadá například následovně:

```
04/08-13:08:04.038614 fe80::b93f:f170:9a36:afb5:51554 -> ff02::c:1900
UDP TTL:1 TOS:0x0 ID:0 IpLen:40 DgmLen:194
Len: 146
```

Pro zobrazení veškerých hlaviček na spojové vrstvě použijeme příkaz: **snort -ve**
Zachycený paket, který je vyobrazen na obrazovce vypadá například následovně:

```
04/08-13:19:04.150159 1C:39:47:D0:DC:4D -> 33:33:00:00:00:0C type:0x86DD len:0
   xD0
fe80::b93f:f170:9a36:afb5:51554 -> ff02::c:1900 UDP TTL:1 TOS:0x0 ID:0
IpLen:40 DgmLen:194 Len: 146
```

Pro zobrazení kompletního paketu, včetně jeho datové části použijeme příkaz: **snort -ved** Zachycený paket, který je vyobrazen na obrazovce vypadá například následovně:

```
04/08-13:20:53.173338 00:15:C7:27:E1:00 -> 68:F7:28:26:9A:85 type:0x800 len:0
   xDC
158.196.0.53:53 -> 158.196.54.226:55218 UDP TTL:64 TOS:0x0
ID:7996 IpLen:20 DgmLen:206 DF Len: 178
5D D5 81 80 00 01 00 02 00 03 00 03 05 64 61 69 ].............dai
73 79 06 75 62 75 6E 74 75 03 63 6F 6D 00 00 01 sy.ubuntu.com...
00 01 C0 0C 00 01 00 01 00 00 00 11 00 04 A2 D5 ................
21 84 C0 0C 00 01 00 01 00 00 00 11 00 04 A2 D5 !...............
21 6C C0 12 00 02 00 01 00 02 30 C6 00 10 03 6E !l........0....n
73 32 09 63 61 6E 6F 6E 69 63 61 6C C0 19 C0 12 s2.canonical....
00 02 00 01 00 02 30 C6 00 06 03 6E 73 33 C0 52 ......0....ns3.R
C0 12 00 02 00 01 00 02 30 C6 00 06 03 6E 73 31 ........0....ns1
C0 52 C0 7C 00 01 00 01 00 02 30 C6 00 04 5B BD .R.|......0...[.
5E AD C0 4E 00 01 00 01 00 02 30 C6 00 04 5B BD ^..N......0...[.
5F 03 C0 6A 00 01 00 01 00 02 30 C6 00 04 5B BD _..j......0...[.
5B 8B
```

4.1.2 Režim logování paketů (Packet logger mode)

Režim logování paketů umožňuje uživateli ukládat pakety zajištěné v režimu Sniffer na pevný disk. Zaznamenává veškeré informace o paketech do databáze, nebo do binárních textových souborů. Správcům sítě jsou pak k dispozici informace například o přesném čase, kdy byl paket pořízen, zdrojové IP adrese, číslu zdrojového portu, cílové IP adrese, číslu cílového portu, protokolu použitého v transportní vrstvě, délce IP záhlaví, délce záhlaví TCP, payload (samotná data) a další.

Prostřednictvím tohoto režimu může uživatel specifikovat pravidla pro ukládání paketů. Může například ukládat pouze pakety vztahující se ke specifické adrese a podobně.

Pro zapnutí logování do souboru stačí pouze za výše zmíněné příkazy z režimu sniffer přidat parametr „-l“ a za něj specifikovat adresář, do kterého chceme data ukládat. V praxi se nejčastěji využívá adresář defaultní /var/log/snort, ale je možnost zvolit i adresář odlišný.
Příkaz: **snort -ved -l /var/log/snort**

Pokud chceme data ukládat ve formátu binárním, stačí pouze na konci přidat ještě parametr „-b“ a data budou ukládány v binární podobě. Binární podoba ukládání dat je vhodná především v sítích s velkým provozem.

4.1.3 Režim detekce narušení sítě (Network intrusion detection mode)

V režimu detekce narušení sítě Snort zaznamenává pouze ty pakety, které odpovídají určitému pravidlu a generuje výstrahy (alerty). Pravidla lze získat ze samotných instalačních souborů a jsou neustále aktualizovány. Tyto výstrahy nabývají různých priorit a pro každou tuto kategorii mohou být přijatá různá opatření. Samotný Snort lze nakonfigurovat tak, aby fungoval v neviditelném režimu a vetřelec přítomnost IDS vůbec nezpozoroval. Jedním ze způsobů, jak toho docílit, může být například nastavení žádné IP adresy na senzorový Snort server.

Snort může na základě pravidel provádět různá rozhodnutí, jakými mohou být například ignorování paketů, zaznamenávání paketů, generování výstrah, aktivování dalších akcí po vygenerování výstrahy, nebo provedení uživatelem definované akce, jako je posílání zpráv do syslogu, odesílání SNMP trapů, logování dat do xml souborů, nebo kombinace těchto akcí. Standardní pravidla Snortu lze měnit podle podmínek dané sítě.

Pro zapnutí NIDS režimu je třeba kromě samotné specifikace výstupního adresáře také specifikovat soubor konfigurační[**?**]. Slouží k tomu parametr „-c“. Příkaz může vypadat takto:
snort -c /usr/local/snort/etc/snort.conf -l /var/log/snort

4.2 Komponenty IDS systému Snort

Snort je logicky rozdělen do následujících pěti komponent. Všechny tyto komponenty pracují společně, aby detekovaly jednotlivé útoky a následně se provádí generování výstupu. Formát výstupu si můžeme nastavit dle svých požadavků v konfiguračním souboru Snortu.

4.2.1 Jednotka paketového záchytu

Jednotka paketového záchytu je první komponenta, která sbírá pakety z různých síťových rozhraní a tyto pakety připravuje pro preprocesory, neboli druhou komponentu Snortu. Má za úkol určit, které základní protokoly jsou v paketu použity, stejně jako určit umístění a velikost paketových dat. Tento v podstatě dekodér také vyhledává anomálie v hlavičkách, které pak mohou způsobit generaci výstrah.

4.2.2 Zásuvné moduly preprocesoru

Tyto moduly se používají k uspořádání a úpravě paketů před jejich analýzou, kterou provádí detekční jednotka. Každá služba má svůj odpovídající preprocesor pro ověření anomálií specifických pro danou službu. Službami mohou být například HTTP nebo FTP. Preprocesory se využívají ale také pro defragmentaci paketu v případě přenášení velkého množství dat.

4.2.3 Detekční jednotka

Třetí komponenta, neboli detekční jednotka je srdcem celého Snortu. Její funkcí je analýza všech paketů, které ji projdou. Analýza se provádí na základě předem nadefinovaných pravidel obsahujících vzory známých útoků. Detekční jednotka může paket rozdělit a pravidla aplikovat jen na určité části paketu, jako například na IP záhlaví, záhlaví transportní vrstvy, záhlaví aplikační vrstvy, nebo na samotná data (užitečné zatížení - packet payload).

4.2.4 Systém logování a výstrah

V případě objevení podezřelého provozu v síti je buďto upozorněn správce sítě, nebo je vygenerována výstraha.

4.2.5 Výstupní zásuvné moduly

Výstupní zásuvné moduly, neboli plug-iny se používají k řízení typu výstupu, vytvořeného systémem logování a výstrah. Těmito funkcemi mohou být například: generování reportů, logování výstražných zpráv do souboru, odesílání SNMP trapů, odesílání zpráv na server Syslog a pod.

4.3 Pravidla Snortu

Stejně jako samotné viry, má i veškerá nebezpečná aktivita nějaké vzory (podpisy). Informace o těchto vzorech slouží k vytváření pravidel pro Snort. Existují databáze známých zranitelností, které Snort využívá pro vytváření pravidel. Tyto vzory mohou být přítomny v části hlavičky paketu, nebo samotných datech (payload) a pravidla lze použít pro kontrolu různých částí datového paketu. Ve verzi Snortu 1.x lze analyzovat záhlaví třetí a čtvrté vrstvy, ale nelze analyzovat protokoly aplikačních vrstev. Podpora analýzy záhlaví aplikačních vrstev však byla přidána ve verzi 2.x. Pravidla mohou být použita pro generování výstražné zprávy, logování do souboru, nebo na obrazovku a jsou napsána snadno pochopitelnou syntaxí. Většina pravidel mají rozsah jednoho řádku, ale mohou být rozšířena na více řádků pomocí znaku zpětného lomítka na konci řádku.

Vlastní pravidla můžeme uspořádat stejně přehledně, jako pravidla implicitní, ale není dobré je spolu míchat. Pro vlastní napsaná pravidla máme připravený soubor local.rules, který můžeme

dále členit. Všechny pravidla pak můžeme zahrnout v konfiguračním souboru snort.conf pomocí klíčového slova „include“. Pro psaní pravidel lze použít online nástroj SNORPY[1].

4.3.1 Struktura pravidel

Veškerá pravidla Snortu mají dvě logické části, záhlaví pravidla a samotné možnosti pravidla.

Obrázek 4: Základní struktura pravidla pro Snort

Hlavička pravidla	Možnosti pravidla

Záhlaví pravidla obsahuje informace o tom, jakou akci má pravidlo provést. Obsahuje také kritéria pro přizpůsobení pravidla pro datové pakety. Samotné možnosti pravidla obvykle obsahují výstražnou zprávu a informace o tom, která část paketu by měla být použitá pro vygenerování výstražné zprávy. V této části jsou obsaženy také další kritéria pro přizpůsobení pravidla pro datové pakety. Pravidla může detekovat jeden, ale i více typů narušení. Správně napsaná pravidla by měla být použitelná pro více vzorů.

Obrázek 5: Struktura hlavičky pravidla pro Snort

Akce	Protokol	Zdrojová adresa	Zdrojový port	Směr	Cílová adresa	Cílový port

Akce: Akce nám říká, co se vlastně má provést v případě, že provoz bude odpovídat danému pravidlu. Akcemi mohou být:

alert – Vygeneruje upozornění a poté je paket uložen do výstupního souboru.

log – Paket bude uložen do výstupního souboru.

pass – Paket bude ignorován.

drop – Paket bude blokován a uložen do výstupního souboru.

reject – Paket bude blokován, uložen a následně se provede TCP reset, pokud se bude jednat o protokol TCP, nebo nastaví ICMP port na unreachable, pokud se bude jednat o UDP.

sdrop – Paket je blokován, ale nikde se neukládá.

Protokol: V současné době Snort podporuje protokoly TCP, UDP, ICMP a IP. V budoucnu budou možná přidány i ARP, IGRP, GRE, OSPF, RIP, IPX atd.

Zdrojová adresa: IP adresa, ze které provoz přichází.

Zdrojový port: Port, ze kterého provoz přichází.

Směr: Směrový operátor označuje orientaci, nebo směr provozu, na který se pravidlo vztahuje. K dispozici je také obousměrný operátor.

Cílová adresa: Cílová IP adresa, na kterou má být provoz směrován.

Cílový port: Cílový port, na který má být provoz směrován.

[1] http://snorpy.com

4.4 Výstupní moduly Snortu

Výstupní moduly byly do Snortu přidány od verze 1.6 a dělají Snort flexibilnějším při formátování a prezentaci výstupů svým uživatelům. V konfiguračním souboru Snortu můžeme povolit více výstupních zásuvných modulů současně. V případě zadání více zásuvných modulů za sebou stejného typu (log, nebo výstraha), jsou tyto moduly volány v pořadí podle toho, kdy daná událost nastane. Výstupní pluginy standardně odesílají svá data do adresáře /var/log/snort, nebo do adresáře, který si nastaví sám uživatel.
Výstupní moduly jsou načítány za běhu zadáním výstupního klíčového slova v konfiguračním souboru:

```
output <name>: <options>
output alert_syslog: log_auth log_alert
```

4.4.1 Alert syslog

Tento modul zasílá výstrahy do syslogu (podobně jako přepínač příkazové řádky -s). Tento modul také umožňuje uživateli specifikovat možnosti logování a prioritu v konfiguračním souboru Snortu, což uživatelům poskytuje větší flexibilitu při zaznamenávání výstrah.
Formát:

```
alert_syslog: \
        <facility> <priority> <options>

output alert_syslog: \
        [host=<hostname[:<port>],] \
        <facility> <priority> <options>
```

4.4.2 Alert fast

Alert fast generuje rychlé upozornění v jednořádkovém formátu do zadaného výstupního souboru. Jedná se o rychlejší metodu pro zaznamenání výstrah, než při ukládání úplné výstrahy, protože nemusí ukládat všechny záhlaví paketů do výstupního souboru a protokolovat pouze jeden soubor.
Formát:

```
output alert_fast: [<filename> ["packet"] [<limit>]]
<limit> ::= <number>[('G'|'M'|K')]
```

filename: Jméno výstupního souboru. Výchozí název je: 2#2logdir3#3/alert. Pro výstup do terminálu lze zvolit možnost „stdout“. Název může obsahovat absolutní, nebo relativní cestu.
packet: Tato volba způsobí, že jsou zaznamenány i víceřádkové položky s úplnými záhlavími paketů. Ve výchozím nastavení jsou zaznamenány pouze jednořádkové položky.

limit: Volitelný limit velikosti souboru, který je standardně 128MB a minimální je pak 1KB.

4.4.3 Alert full

Tento typ výstrahy ukládá všechny záhlaví paketů a z toho důvodu je ukládání zdlouhavé.
Formát:

```
output alert_full: [<filename> [<limit>]]
    <limit> ::= <number>[('G'|'M'|K')]
```

filename: Jméno výstupního souboru. Výchozí název je: 2#2logdir3#3/alert. Pro výstup do terminálu lze zvolit možnost „stdout“. Název může obsahovat absolutní, nebo relativní cestu.
limit: Volitelný limit velikosti souboru, který je standardně 128MB a minimální je pak 1KB.

4.4.4 Alert unixsock

Alert unixsock nastaví doménu UNIX a zašle do ní všechny výstražné zprávy. Externí programy a procesy mohou v tomto soketu naslouchat a přijímat výstražná paketová data v reálném čase.
Formát:

```
alert_unixsock
```

4.4.5 Log tcpdump

Modul log tcpdump zaznamenává pakety do souboru ve formátu tcpdump. Tento typ výstupu je vhodný pro provádění následné analýzy shromážděného provozu velkým počtem nástrojů, které jsou k dispozici pro soubory ve formátu tcpdump.
Formát:

```
output log_tcpdump: [<filename> [<limit>]]
    <limit> ::= <number>[('G'|'M'|K')]
```

filename: Jméno výstupního souboru. Výchozí název je: 2#2logdir3#3/snort.log. Název může obsahovat absolutní, nebo relativní cestu. K názvu souboru je připojeno časové razítko UNIX.
limit: Volitelný limit velikosti souboru, který je standardně 128MB.

4.4.6 csv

Zásuvný modul csv umožňuje zápis výstražných dat ve formátu snadno importovatelném do databáze. Výstupní položky a jejich pořadí lze přizpůsobit.

Formát:

```
output alert_csv: [<filename> [<format> [<limit>]]]
<format> ::= "default"|<list>
<list> ::= <field>(,<field>)*
<field> ::= "dst"|"src"|"ttl" ...
<limit> ::= <number>[('G'|'M'|K')]
```

filename: Jméno výstupního souboru. Výchozí název je: 2#2logdir3#3/alert.csv. Pro výstup do terminálu lze zvolit možnost „stdout“. Název může obsahovat absolutní, nebo relativní cestu.
format: Možnosti formátování.
limit: Volitelný limit velikosti souboru, který je standardně 128MB a minimální je pak 1KB.

4.4.7 Unified 2

Unified 2 může pracovat v jednom ze tří režimů. Těmito režimy jsou: logování paketů, logování výstrah a kombinace obou předchozích. Logování paketů zahrnuje zachycení celého paketu a specifikuje se v konfiguračním souboru pomocí „log_unified2“. Logování výstrah zaznamenává pouze události a v konfiguračním souboru se specifikuje pomocí „alert_unified2“. Pro zahrnutí obou těchto způsobů pro logování do jednoho souboru, použijeme pro specifikaci jednoduše „unified2“.

Při zapnuté podpoře MPLS mohou být do Unified2 události zahrnuty také MPLS štítky. Pro povolení MPLS lze použít volbu „mpls_event_types“. Bez této volby nebudou informace o MPLS ukládány.
Formát:

```
output alert_unified2: \
    filename <base filename> [, <limit <size in MB>] [, nostamp]
    [, mpls_event_types] \ [, vlan_event_types]

output log_unified2: \
    filename <base filename> [, <limit <size in MB>] [, nostamp]

output unified2: \
    filename <base file name> [, <limit <size in MB>] [, nostamp]
    [, mpls_event_types] \ [, vlan_event_types]
```

5 Grafická a webová rozhraní pro Snort

Pro IDS systém Snort existuje řada grafických a webových rozhraní. Tyto rozhraní se od sebe liší například v tom, jaký programovací jazyk byl použit při jejich vývoji, jakým způsobem události zpracovávají, jakým způsobem provádí analýzu nad daty a podobně. K podrobnějšímu popisu a porovnání jsem si vybrala následující rozhraní.

5.1 SnortAlog

SnortAlog je vlastně skript napsaný v Perlu a umožňuje zobrazit útoky na síť detekovány Snortem. Může generovat grafy v HTML, PDF a textové výstupy. SnortAlog funguje se všemi verzemi Snortu a dokáže analyzovat výstupy ve formátu alert syslog, alert fast a alert full [15].
Hlavními možnostmi, které SnortAlog nabízí jsou:

- Vytváření HTML, PDF a textové výstupy s kódováním ASCII.
- Dokáže řadit události vzestupně a sestupně.
- Nabízí možnost určit počet zobrazených událostí.
- Dokáže vyřešit IP adresy a domény.
- Nabízí napojení na informace z databáze WhoIs.
- Nabízí grafické uživatelské rozhraní.
- Nabízí možnost provádět filtrovaní například podle protokolu a pod.
- Generuje GIF, PNG nebo JPG grafy ve výstupu HTML.

Dále je SnortAlog schopen pracovat se Snortem a nabízí například:

- Analýzu, kterou lze vytvářet ze tří různých formátu výstupu.
- Pracuje se všemi preprocesory (spp_stream4, spp_portscan, spp_decoder, flow a flow-portscan ...).
- Propojení vzoru s popisem útoku pomocí webového odkazu.
- Pracuje s volbou „-I“ (specifikace rozhraní a přidání reportů).
- Pracuje s volbou „-e“ (zobrazení informace o hlavičce druhé vrstvy).
- Použití specifického pluginu pro generování vlastních referenčních pravidel.

Podporovanými operačními systémy jsou: Linux, FreeBSD, OpenBSD, Solaris, Windows, MacOS, Fedora, CentOS
Poslední stabilní verze: 2.4.3

5.2 Base

Base prohledává a zpracovává databázi obsahující události zaznamenané nejrůznějšími nástroji pro monitorování sítě, jako jsou brány firewall a IDS systémy. Base je napsán v programovacím jazyce PHP a informace zobrazuje formou webového rozhraní. V kombinaci se Snortem base dokáže zpracovat binární formáty tcpdump a všechny formáty výstrah. Graficky zobrazuje informace o paketech třetí a čtvrté vrstvy. Generuje také grafy a statistiky založené na čase, senzoru, podpisu, protokolu, IP adrese, portu TCP/UDP, nebo klasifikaci. Base umožňuje také vyhledávání v událostech na základě meta informací o výstrahách, jako je senzor, skupina výstrah, podpis, klasifikace a doba detekce, jakož i paketová data, jako jsou adresy, zdrojové a cílové porty, užitečné pakety, nebo příznaky paketů.

Base umožňuje správu událostí. Správce může kategorizovat data do skupin, smazat falešně pozitivní, nebo dříve zpracována upozornění a archivovat a exportovat data na e-mailovou adresu pro administrativní oznámení, nebo pro další zpracování. Podpora přihlašování uživatelů a rolí umožňuje administrátorovi kontrolovat výsledný obsah vyobrazený pomocí webového rozhraní.

Jelikož ale Snort od verze 2.9.2 již nemá podporu výstupu do databáze, tak je třeba použití dalšího softwaru pro převod dat. Pro tyto účely se používá Barnyard. Nutnost Barnyardu považuji za nedostatek, jelikož je potřeba si veškeré data nejdříve převést do databáze a až pak je možnost jejich zobrazení pomocí rozhraní base. Pokud chceme mít data aktuální, tak je nutnost mít neustále tento proces převodu dat zapnut v pozadí pomocí příkazu v příkazové řádce a jelikož tento proces běží neustále, tak může způsobovat zpomalení zařízení, na kterém je vše spuštěno. Dále je třeba mít k dispozici webový server, jakým je například Apache.

Podporovanými operačními systémy jsou: Linux, Windows, FreeBSD, OpenBSD, Fedora, CentOS

Poslední stabilní verze: 1.4.5

5.3 AlienVault OSSIM

AlienVault OSSIM je volně dostupný, ale je podporován dalšími komerčními subjekty. OSSIM nabízí různé funkce, jako například kolekce událostí, normalizace a korelace. Společnost AlienVault OSSIM byla založena bezpečnostními inženýry kvůli nedostatku dostupných produktů s otevřeným zdrojovým kódem.

OSSIM má čtyři hlavní složky: senzor, databázi, framework a server. Senzor připojuje zabezpečovací zařízení a server pro správu. Senzory používají pluginy k analýze dat z bezpečnostních zařízení. Serverem pro správu je server OSSIM a jeho nasazení je jednoduché. Má jeden server, ale ve složitějších prostředích, kde je vyžadována vysoká dostupnost, existuje mnoho strategicky umístěných serverů s různými rolemi. OSSIM nabízí uživatelské rozhraní ve formě webové aplikace, ale i prostřednictvím konzole. K ukládání události je ale třeba MySQL databáze, tímto narážíme na stejný problém s Barnyardem jako tomu bylo u webového rozhraní BASE. OSSIM také postrádá pokročilejší funkce dlouhodobého ukládání dat.

OSSIM poskytuje jednu jednotnou platformu s mnoha základními bezpečnostními schopnosti jako například:

1. Zjišťování aktiv.
2. Posuzování zranitelnosti.
3. Detekce narušení.
4. Monitorování chování provozu.
5. Korelace události SIEM (Security Information and Event Management).

Podporovanými operačními systémy jsou: Windows, Mac OS, Linux, Fedora, FreeBSD, OpenBSD
Poslední stabilní verze: v5.7.1

5.4 Sguil

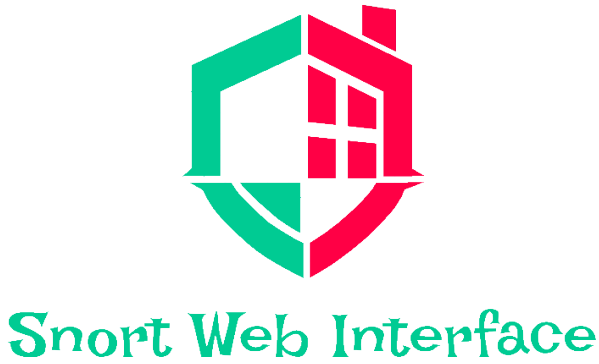
Sguil slouží pro analýzu síťové bezpečnosti a nabízí grafické uživatelské rozhraní. Sguil se skládá z jednoho Sguil serveru a libovolného počtu Sguil síťových senzorů. Senzory pravidelně provádějí všechny úlohy monitorování zabezpečení a výsledky vrací serveru. Server tyto informace zpracovává, ukládá do databáze a komunikuje s klienty běžícími na počítačích administrátorů. Klient Sguil je napsán v tcl / tk a může být spuštěn na jakémkoliv operačním systému, který podporuje tcl / tk. Sguil pro svou funkčnost potřebuje Barnyard, který bere události z výstupního souboru Snortu a odesílá je agentovi senzoru, který je vkládá do databáze spuštěné na Sguil serveru. Barnyard však také představuje již výše zmíněné nevýhody.
Podporovanými operačními systémy jsou: Linux, FreeBSD, OpenBSD, Mac OS, Solaris, Fedora, CentOS
Poslední stabilní verze: 0.8.0

	Linux	Windows	Mac OS	FreeBSD	OpenBSD	Solaris	Fedora	CentOS	Poslední verze
SnortAlog	✓	✓	✓	✓	✓	✓	✓	✓	2.4.3
Base	✓	✓		✓	✓		✓	✓	1.4.5
AlienVault OSSIM	✓	✓	✓	✓	✓		✓		v5.7.1
Sguil	✓	✓	✓	✓	✓	✓	✓	✓	0.8.0

Tabulka 2: Srovnání grafických rozhraní pro Snort

6 Grafické rozhraní vytvořené v praktické části práce

Cílem této diplomové práce bylo vytvořit webové rozhraní pro Snort, které bude umožňovat přehlednou vizualizaci zachycených útoků s možností stáhnout zachycené paketové data. Vlastní implementované rozhraní jsem pojmenovala jako SnortWebInterface.



Obrázek 6: Logo SnortWebInterface

6.1 Platforma

Jako platformu jsem si zvolila moderní a pro mne velmi atraktivní programovací jazyk **Ruby On Rails**. Rails je framework pro vývoj webových aplikací v jazyce Ruby využívající architekturu MVC. Základní myšlenkou architektury MVC je oddělení logiky od výstupu. Zjednodušeně řečeno MVC dělí aplikaci na 3 logické části tak, aby šlo jednotlivé části samostatně upravovat s co nejmenším dopadem na části ostatní. Těmito částmi jsou: modely, pohledy a kontroléry.

Modely obsahují většinu aplikační logiky. Reprezentují nám data, ale také pravidla práce s těmito daty. V Railsech slouží modely především k interakci s danou tabulkou v databázi a pro ukládání pravidel této interakce. Mezi pravidla ukládání mohou patřit například pravidla validační. Validace je kontrola správnosti dat před jejich uložením v databázi. Většinou odpovídá jedna tabulka v databázi jednomu modelu v aplikaci.

Pohledy nám upravují výsledné uživatelské rozhraní naší aplikace. Mohou to být jak prvky HTML s částmi Ruby kódu, tak také novější Haml. V pohledech můžeme také volat různé javascriptové kódy.

Kontroléry fungují zjednodušeně řečeno jako „lepidlo“ mezi modely a pohledy. V Railsech kontroléry vlastně získávají data z modelů a odesílají je do pohledů, kde jsou zobrazovány. V kontrolérech můžeme tedy upravovat to, jaké data budeme v pohledech vyobrazovat. Můžeme k tomu používat například různé podmínky. Musíme si ale dávat pozor, aby kontroléry neobsahovaly příliš mnoho kódu, proto si vytváříme různé pomocné metody a servisy.

6.2 Použité technologie

Pro tvorbu rozhraní bylo použito množství různých technologií pro usnadnění práce. Především se jedná o ruby gemy.

6.2.1 Gemy

Gemy jsou vlastně balíčkovací systémy pro Ruby, umožňující instalaci knihoven. Jelikož jsou gemy napsány v Ruby, tak jsou také multiplatformní, to znamená, že jsou snadno přenositelné mezi různými platformami. Mezi nedůležitější gemy, které byly použity při tvorbě rozhraní patří gemy následující.

6.2.1.1 Devise – Devise je flexibilní autentizační řešení pro Ruby On Rails. Devise se v aplikaci stará o registraci nových uživatelů, přihlašování a odhlašování uživatelů, tvorbu nového hesla při zapomenutém hesle a pod. Umožňuje také současné přihlášení více uživatelů. Hesla se neukládají do databáze přímo, ale ukládá se pouze jejich „heš“. Přihlašování funguje formou ukládání „sešny“ do souboru cookies (cookies jsou krátké textové soubory, které vytváří webový server a ukládají se v počítači prostřednictvím prohlížeče. Prohlížeč pak při příští návštěvě zašle uloženou cookie zpět na server a získá tak všechny dříve uložené informace).

6.2.1.2 Ipaddress – Tento gem jsem v aplikaci použila pro správný převod IPv6 adres. Snort ukládá IPv6 adresy jako celá čísla bez znaménka o velikosti 128 bitů, je tedy třeba provést převod do klasické podoby IPv6 adresy, která je v hexadecimálním tvaru.

```
def source_ip
  if ip_source.length == 38
    u128 = ip_source.to_i
    address = IPAddress::IPv6::parse_u128 u128
  else
    address = ip_source
  end
  address.to_s
end
```

Výpis 1: Ukázka řešení pro IPv6 adresy

6.2.1.3 Geocoder – Tento gem dokáže určit adresu a souřadnice na základě IP adresy. Používám ho pro získání více informací ze zdrojové IP adresy. Volám ho v aplikaci pro každou událost pouze jednou a výsledky si ukládám do své databáze. Část kódu, který využívá tento gem můžete vidět v ukázce 2.

```
#info from source ip
info_from_source_ip = Geocoder.search(self.source_ip)
if info_from_source_ip.present?
  self.lat = info_from_source_ip.first.coordinates.first
  self.lng = info_from_source_ip.first.coordinates.last
  self.country = info_from_source_ip.first.country
  self.full_address = info_from_source_ip.first.address
end
```

Výpis 2: Ukázka použití gemu Geocoder

6.2.1.4 WhoIs – Gem WhoIs slouží pro získání dalších informací z IP adresy a to ze serveru WhoIs. WhoIs je schopen zpracovat IP adresy verze 4 i 6 a výstupem jsou informace o tom, v jakém bloku se adresa nachází a jaká organizace má tuto IP adresu přidělenou. WhoIs volám v aplikaci pouze jednou a výsledky si ukládám do své databáze a to hned za voláním Geocoderu.

```
#whois info
whois_from_source_ip = Whois::Client.new
whois_from_destination_ip = Whois::Client.new

self.whois_info_from_source_ip = whois_from_source_ip.lookup(self.source_ip)
self.whois_info_from_destination_ip = whois_from_destination_ip.lookup(self.
    destination_ip)
```

Výpis 3: Ukázka použití gemu WhoIs

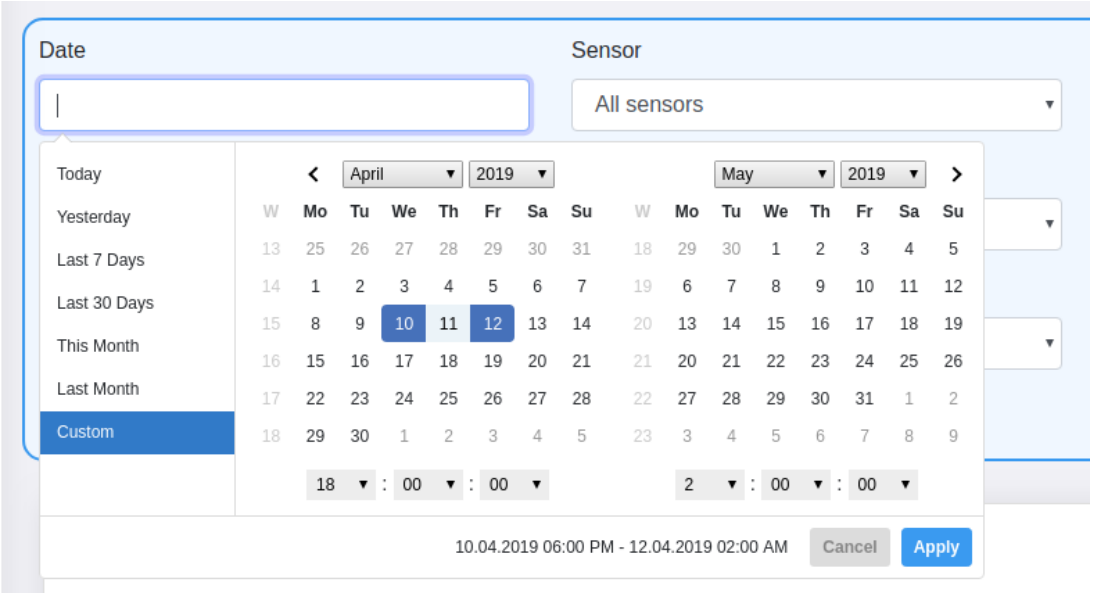
6.2.1.5 Bootstrap – Bootstrap jsem využila jakožto nejoblíbenější HTML, CSS a javascriptovou knihovnu na světě. Bootstrap nabízí podporu responzivity. Responzivní webová aplikace je taková aplikace, která dokáže veškeré prvky na stránce přizpůsobit v závislosti na velikosti displeje, na kterém je zobrazena. Znamená to tedy, že je aplikace přehledná například i na mobilním telefonu a tabletu.

6.2.1.6 Unified2 – Jakožto formát výstupního souboru pro následnou analýzu jsem si vybrala Unified2 formát zahrnující jak samostatné události, tak také pakety. Gem Unified2 dokáže číst soubory tohoto typu a v aplikaci je k tomuto účelu také využit.

6.2.1.7 Carrierwave – Carrierwave poskytuje jednoduché a flexibilní řešení ukládání souborů do Ruby On Rails aplikace. Používám ho pro ukládání souboru s událostmi v případě, že

chce uživatel události načítat touto externí metodou. Používám ho také pro ukládání avataru uživatele.

6.2.1.8 Bootstrap daterangepicker – Bootstrap daterangepicker je komponenta, která velice usnadňuje výběry časových úseků. Můžeme si zvolit dnešní den, včerejší den, posledních sedm dní, posledních třicet dní, tento měsíc, minulý měsíc, nebo si sami zvolit data. Pokud si data pro vyplnění daného vstupního pole volíme sami, volíme si i časy. V aplikaci tuto komponentu používám pro vstupní pole výběru časového úseku při filtrování událostí.



Obrázek 7: Daterange picker

6.2.1.9 Annotate – Annotate sice nijak neovlivňuje chod aplikace, ale v aplikaci slouží pro přehlednější orientaci ve vytvořené databázi. U každého modelu nám vypíše kompletní tabulku v databázi vztahující se k danému modelu, což dělá kód aplikace přehlednějším.

6.2.1.10 Haml – Haml je šablonovací nástroj pro HTML. Je navržen tak, aby usnadňoval a zpříjemňoval psaní HTML dokumentů tím, že eliminuje nadbytečnost kódu, což se odráží v základní struktuře, kterou dokument představuje. Haml v aplikaci poskytuje elegantní syntaxi, která je výkonnější a přehlednější, než syntaxe HTML kódu.

6.2.1.11 Turbolinks – Turbolinks umožňuje rychlejší procházení v aplikaci. Tato funkce v podobě gemu je již v nových Rails aplikacích standardně povolena. Turbolinks v aplikaci funguje tak, že zachycuje veškeré kliknutí na odkazy, které navigují na další stránku v aplikaci a místo aby překreslil celou stránku, tak tuto žádost vyřeší prostřednictvím AJAXu, který dokáže nahradit jen samotné tělo stránky přijatým obsahem.

6.2.2 Docker

Jelikož je aplikace napsána v Ruby On Rails a její instalace by mohla být příliš složitá, tak jsem se rozhodla využít pro její běh Docker. Docker funguje jako virtuální stroj a izoluje procesy běžící v Dockeru od procesů ostatních. Docker využívá tzv. kontejnéry, ve kterých běží služby. Start tohoto kontejneru (virtuálního stroje) je okamžitý a nedochází tedy k zatížení paměti RAM, ani ke zpomalení při vlastním běhu programu. V RoR je pro správné fungování Dockeru potřeba dvou souborů. Prvním je Dockerfile, což je jednoduchý textový soubor popisující výslednou image. Dále je třeba konfigurace souboru docker-compose.yml, který se věnuje samotným službám, které jsou potřeba pro správnost fungování aplikace. V rozhraní vytvořeném v praktické části jsou tyto služby celkem tři. První kontejner je pro Redis server. Redis funguje jako úložiště dat ve struktuře paměti. Redis podporuje datové struktury jako řetězce, heše, seznamy, bitmapy a mnoho dalšího. Druhý kontejner běží pro PostgreSQL databázi a v posledním kontejneru běží samotná aplikace. Při startu Dockeru se z posledního kontejneru spouští také skript s názvem „start“, ve kterém provádím další potřebné instalace. Instaluji zde například knihovny libpq-dev, nodejs, libpcap-dev a ruby-pcaprub, které jsou potřeba pro správnou funkci gemů. Následně se instalují veškeré gemy. Při prvním spuštění se v tomto skriptu také vytvoří databáze a provede se migrace. Migrace znamená vytvoření databázové struktury podle předem nadefinovaných parametrů. Nakonec skript zkontroluje, zda server již neběží jinde a pokud ne, tak server spustí na adrese localhostu a portu 3000.

6.3 Instalace aplikace

Jelikož je použit výše zmíněný Docker, tak je instalace aplikace velice snadná. Pro spuštění je potřeba mít nainstalován Docker Compose.

Pokud máme Docker Compose nainstalován správně, tak je již spuštění aplikace velice snadné. Přesuneme se v terminálu do složky, kde je aplikace umístěna. Při prvním spuštění je třeba zavolat příkaz **sudo docker-compose build** , který slouží k vytvoření kontejnérů. Pokud vše proběhne v pořádku, můžeme spustit již samotnou aplikaci a to pomocí příkazu **sudo docker-compose up**. První spuštění tohoto příkazu trvá o něco déle, jelikož se provádí prvotní instalace, nikdy to však netrvá déle, než 2 a půl minuty. Při opětovném spuštění se aplikace spustila už vždy do 10 sekund. Spuštěná aplikace běží na localhostu a portu 3000.

- **http://localhost:3000**

Jelikož aplikace ve verzi 1.0 pracuje pouze s jedním souborem, tak je v případě, že potřebujete zpracovat více souborů najednou , nutné provést sloučení výstupních Unified2 souborů do jednoho a tento soubor pak následně vložit do aplikace pro následnou analýzu.

6.4 Databáze

Rozhraní využívá svou databázi, kterou jsem sama navrhla. Databáze je typu PostgreSQL. Z hlediska databázového modelu se jedná o relační model dat. Tento model je nejmladším a také nejpoužívanějším databázovým modelem. Struktura tohoto modelu je jednoduchá, data jsou organizována v tabulkách a tyto tabulky se dále skládají z řádků a sloupců. Veškeré databázové operace jsou prováděny v těchto tabulkách. Databáze obsahuje celkem pět tabulek a každá tabulka odpovídá modelu v aplikaci.

Struktura databáze je stromová a na jejím vrchole je samotný uživatel (User). Bez vytvoření uživatele se v aplikaci nedostaneme k dalším krokům. Povinnými položkami pro vytvoření nové registrace jsou **email** a **heslo**, bez vyplnění těchto dvou položek registraci nelze provést. Dále může uživatel vložit svou fotografii (**avatar**) pomocí gemu Carrierwave [2] a **uživatelské jméno**. Tabulka uživatele obsahuje více sloupců, tyto sloupce nevyplňuje samotný uživatel, ale aplikace. Těmito sloupci jsou například **sign_in_count** pro celkový počet přihlášení, **last_sign_in_at** pro uchování posledního data přihlášení, **current_sign_in_ip** pro aktuální IP adresu uživatele, **last_sign_in_ip** pro poslední IP adresu uživatele při předchozím přihlášení a admin pro možnosti dalšího vývoje a administrátorské role.

Druhou tabulkou v databázi je tabulka nastavení (Setting). Nastavení má vazbu na uživatele typu 1:1, což znamená, že každý uživatel má pouze jedno nastavení. Dokud uživatel tuto tabulku nevytvoří, tak bude neustále vyzýván k tomu, aby ji vytvořil, jelikož bez uloženého nastavení nelze načíst události pro následnou analýzu a grafické zpracování. Tabulka nastavení obsahuje buď soubor typu Unified2 uložený ve sloupci **logs**, nebo cestu k souboru uloženém ve sloupci **logs_path**. Tabulka ještě navíc obsahuje sloupec **internal** a je typu boolean, což znamená, že bude nabývat jen hodnoty true (ano), nebo false (ne). V závislosti na tom, zda je zvolena interní možnost načítání dat, či nikoliv probíhá následná validace,neboli kontrola správnosti souboru. Aplikace přijímá pouze soubory ve tvaru „snort.u2.*“.

Třetí tabulkou je tabulka nebezpečných událostí (AcidEvent), která již obsahuje konkrétní události načtené ze zvoleného souboru. Tato tabulka je vázaná na uživatele a tato vazba je typu 1:N, což znamená, že uživatel může načíst do aplikace mnoho různých událostí. Jelikož je tato tabulka spolu se škodlivými pakety ze všech tabulek nejpodstatnější, budu se více věnovat oběma těmto tabulkám a jejich sloupcům podrobněji. Předtím je ale třeba představit datové typy, které jsou v databázi používány. Těmito datovými typy jsou:

1. integer – Tento datový typ se používá pro čísla do velikosti čtyř bajtů.
2. string –Tento datový typ se používá pro řetězce.
3. boolean – Tento datový typ nabývá pouze hodnot true a nebo false, neboli ANO/NE.
4. datetime – Tento datový typ se používá pro ukládání data i s přesným časem.

[2] Gem Carrierwave, detailní popis naleznete v sekci 6.2.1.7

Sloupce v tabulce obsahující škodlivé události jsou následující:
sensor – sloupec typu string obsahující název senzoru
timestamp – sloupec typu datetime obsahující datum a přesný čas, kdy byla podezřelá aktivita zachycena
classification – sloupec typu string obsahující samotnou klasifikaci škodlivé události
severity – sloupec typu integer obsahující úrovně závažnosti přiřazené prioritní značkou. Tento stupeň závažnosti může nabývat hodnot 1 až 4
signature – sloupec typu string obsahující vzory známých útoků, které byly přiřazeny k dané škodlivé události
country sloupec typu string obsahující kód země útočníka
ip_source – sloupec typu string obsahující zdrojovou IP adresu útočníka
ip_destination – sloupec typu string obsahující cílovou IP adresu
source_port – sloupec typu integer obsahující zdrojový port, ze kterého provoz přicházel
destination_port – sloupec typu integer obsahující cílový port, na který byl provoz směrován
lat – sloupec typu string obsahující souřadnice zeměpisné šířky získané ze zdrojové IP adresy pomocí Geocoderu [3]
lng – sloupec typu string obsahující souřadnice zeměpisné délky získané ze zdrojové IP adresy pomocí Geocoderu
full_address – sloupec typu string obsahující kompletní adresu útočníka získanou pomocí informací ze zdrojové IP adresy pomocí Geocoderu
has_extras – sloupec typu boolean a říká nám, zda událost obsahuje nějaké extra informace, nebo nikoliv
has_packets – sloupec typu boolean a říká nám, zda událost obsahuje nějaké škodlivé pakety, nebo nikoliv
icmp – sloupec typu boolean a říká nám, zda se událost týká protokolu ICMP, nebo nikoliv
tcp – sloupec typu boolean a říká nám, zda se událost týká protokolu TCP, nebo nikoliv
udp – sloupec typu boolean a říká nám, zda se událost týká protokolu UDP, nebo nikoliv
checksum – sloupec typu string obsahující kontrolní součet
length – sloupec typu integer obsahující délku událostí
microseconds – sloupec typu string obsahující délku události v mikrosekundách
packet_time – sloupec typu datetime obsahující datum a přesný čas zapsání paketu do Unified2 souboru
potocol – sloupec typu string obsahující protokol zachyceného provozu
whois_info_from_source_ip – sloupec typu string obsahující výpis ze serveru WhoIs obsahující rozsáhlé informace získané z IP adresy útočníka
whois_info_from_destination_ip – sloupec typu string obsahující výpis ze serveru WhoIs obsahující rozsáhlé informace získané z cílové IP adresy

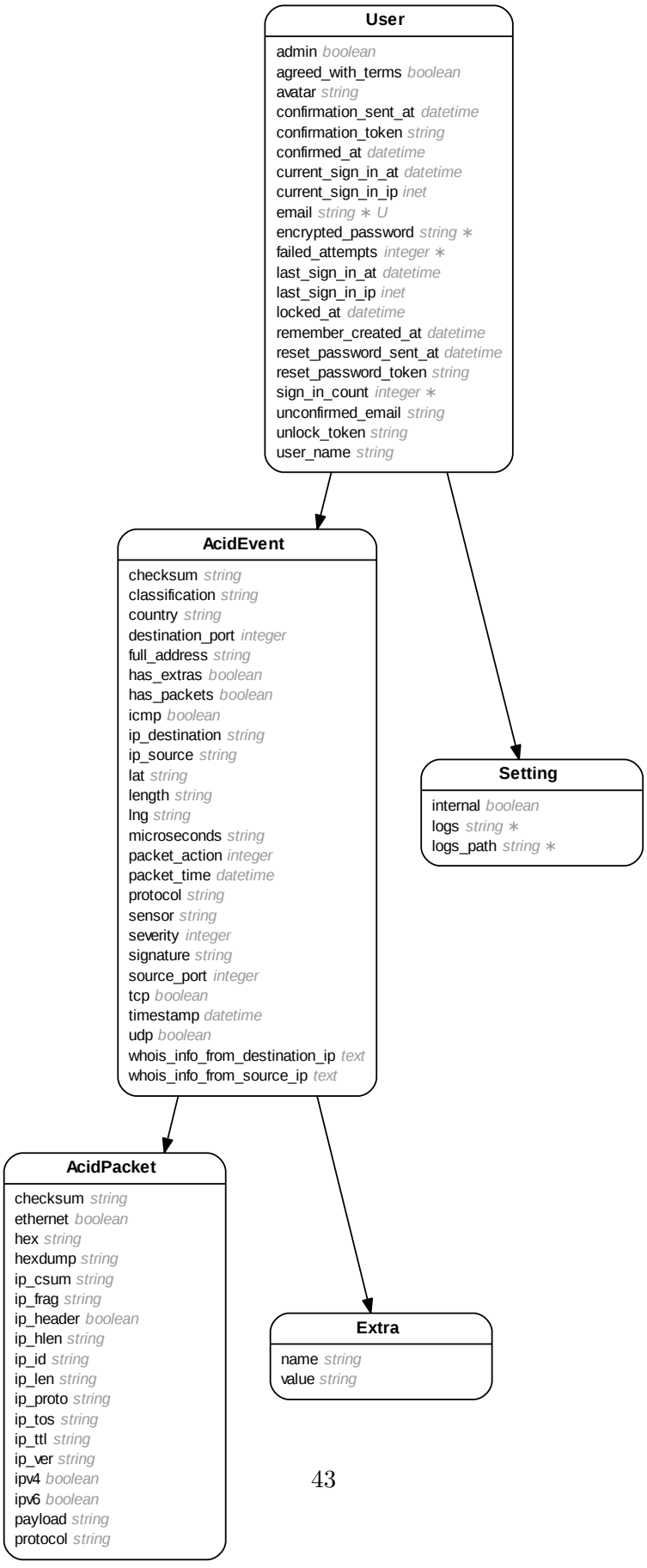
[3] Gem Geocoder, detailní popis naleznete v sekci 6.2.1.3

Čtvrtou tabulkou je tabulka obsahující škodlivé pakety (AcidPacket) a je typu 1:N vůči škodlivým událostem. Tento typ vazby byl zvolen jelikož škodlivá událost může obsahovat vícero paketů najednou.
Sloupce v tabulce obsahující škodlivé pakety jsou následující:
checksum – sloupec typu string obsahující kontrolní součet paketu
ethernet – sloupec typu boolean a říká nám, zda se jedná o provoz přicházející z ethernetu, nebo nikoliv
hex – sloupec typu string obsahující zápis v šestnáctkové soustavě (hexadecimální zápis)
hexdump – sloupec typu string obsahující zápis dat, kdy každý byte je vypsán jako dvojice hexadecimálních číslic. V tomto způsobu zápisu dat lze vidět ordinální ASCII hodnoty zapsány jako šestnáctková čísla
ipv4 sloupec typu boolean a říká nám, zda se jedná o IP adresu verze 4 nebo nikoliv
ipv6 – sloupec typu boolean a říká nám, zda se jedná o IP adresu verze 6 nebo nikoliv
payload – sloupec typu string obsahující samotná data paketu
protocol – sloupec typu sting obsahující informace o použitém protokolu
Následující sloupce obsahují informace získané z IP hlavičky paketu:
ip_header – sloupec typu boolean a říká nám, zda jsou dostupné informace z hlavičky IP, nebo nikoliv
ip_csum – sloupec typu string obsahující kontrolní součet IP hlavičky
ip_frag – sloupec typu string obsahující hodnotu v jednotkách 8 oktetů (64 bitů), která určuje hodnotu pro každý fragment dat v procesu opětovného sestavení
ip_tos – sloupec typu string obsahující informace o typu služby
ip_proto – sloupec typu string obsahující informace o protokolu
ip_ver – sloupec typu integer obsahující číslo verze IP
ip_hlen – sloupec typu string obsahující délku IP hlavičky
ip_len – sloupec typu string obsahující délku IP adresy
ip_id – sloupec typu string obsahující id IP hlavičky
ip_ttl – sloupec typu string obsahující informace o TTL, což je v informatice číslo, které omezuje dobu platnosti dat, nebo počet průchodů paketů skrz aktivní prvky počítačové sítě

Tabulka pátá se vytváří pouze v případě, že událost obsahuje nějaké extra informace. Je připojena k událostem vazbou 1:N jelikož jedna událost může obsahovat i více extra dat. Tato tabulka obsahuje pouze sloupce s názvy a sloupce se samotnou hodnotou.

Vytváření databáze probíhá při spuštění Dockeru ve skriptu start a to v případě, že databáze nebyla již dříve vytvořena.

Obrázek 8: Databázový diagram



6.5 Vzhled webové aplikace

Vzhled výsledného rozhraní jsem se na rozdíl od již existujících rozhraní snažila udělat modernějším, aby odpovídal standardům 21. století. Inspirovala jsem se moderními šablonami určenými pro administrátorské stránky, ale žádná z nich nebyla pro aplikaci použita. Design aplikace je tedy unikátní. Zdrojové kódy se nachází ve složce /assets/stylesheets a využívají moderní technologii SASS, což je nejmodernější rozšíření jazyka CSS. Nechybí také responzivita vytvořená pomocí Bootrapu[4] a vysoká rychlost načítání stránek dosažena pomocí Turbolinks[5].

6.6 Funkce

Webové rozhraní, které jsem vytvořila v praktické části této práce nabízí mnoho různých funkcí. Podrobnějším popisem těchto funkcí se zabývám níže.

6.6.1 Možnost více uživatelů

Rozhraní vytvořené v praktické části vyžaduje po uživateli registraci. Vytváří si tedy databázi uživatelů a není nastaveno žádné omezení pro jejich počet. Povinnými položkami pro úspěšné dokončení registrace jsou emailová adresa a heslo, ale pokud má uživatel zájem, může si nastavit také svůj obrázek (avatara) a uživatelské jméno. Díky využití Devise[6] se do databáze ukládají také další položky, jako například počet přihlášení, datum posledního přihlášení nebo IP adresa posledního zařízení odkud se uživatel přihlásil. Tyto informace se dají velmi dobře využít pro možnosti dalšího vývoje. Pro další možnosti vývoje jsem také již vytvořila administrátorskou roli pro uživatele.

6.6.2 Parsování výstupního souboru a ukládání událostí do databáze.

Nejdůležitější části celé aplikace je bezesporu část věnující se načítání samotných dat. Před importem dat je potřeba se nejdříve zaregistrovat a v nastavení zvolit jednu, ze dvou možností načítání dat. První možnost pro načítání dat využijeme především ve chvíli, kdy na stejném stroji, kde chceme spouštět rozhraní, běží také Snort.

Pro volbu první možnosti vyplníme v nastavení cestu k souboru typu Unified2, který obsahuje události, které chci zpracovávat. Druhou možnost zase využijí ti, kteří mají jen soubor s výstupem ve formátu Unified2 a chtějí provést analýzu, ale Snort na daném stroji neběží. V tomto případě pak stačí tento soubor uložit do aplikace a data se zpracují. Z kontrolních důvodu aplikace pracuje pouze se soubory s názvem začínajícím „snort.u2.*“.

Po uložení nastavení aplikace přechází k již samotnému importu událostí do databáze. Pro tuto funkci je volaná pomocná metoda „parse_events“, která využívá gem Unified2[7] pro pomoc

[4] Gem Bootstrap, detailní popis naleznete v sekci 6.2.1.5
[5] Gem Turbolinks, detailní popis naleznete v sekci 6.2.1.11
[6] Gem Devise, detailní popis naleznete v sekci 6.2.1.1
[7] Gem Unified2, detailní popis naleznete v sekci 6.2.1.6

s přečtením souboru. Veškeré události jsou vázány na uživatele a ostatní je tedy nemohou vidět. Na řádku 2. můžeme vidět, že pro zavolání metody „parse_events“ je třeba předat konkrétního uživatele, který je zrovna do aplikace přihlášen. Každý uživatel může načítat z libovolného souboru a po přihlášení uvidí pouze své načtené události. V metodě „parse_events“ je obsažena také kontrola, zda již v databázi neexistuje záznam s odpovídajícím časovým razítkem a pouze v případě, že žádný takový záznam neexistuje, pokračuje metoda dále. Toto se provádí na řádku 9. a je využita metoda vytvořena v modelu AcidEvent, pomocí které dochází k rychlému hledání v tabulce událostí v závislosti na přihlášeném uživateli .V dalším kroku se vytvoří nová událost a pomalu se do nově vytvořené události přiřazují relevantní data což můžeme pozorovat v ukázce kódu od řádku 11. Na konci metody se událost uloží. Po naimportování všech událostí ze souboru jsme přesměrování na hlavní stránku aplikace a vidíme zde hlášení, že vše proběhlo v pořádku.

V případě, že máme zvolenou možnost vkládaní událostí pomocí zadané cesty k souboru a v tomto souboru přibyly nové události, které bychom chtěli načíst, stačí provést pouze aktualizaci. Aktualizaci lze provést na hlavní stránce s výpisem všech událostí. Aktualizace funguje tak, že se znovu spustí celý proces načítání událostí a pokud metoda narazí na událost jejichž časové razítko zatím databáze neobsahuje, vytvoří ji. Na konci procesu jsme informování, zda byly vytvořeny nové události, nebo zda byl soubor neměnný.

Kdykoliv máme také možnost nastavení změnit. Při změně nastavení se provádí kontrola, zda se soubor pro načítání událostí skutečně změnil. Tato kontrola spočívá v porovnávání hešů. Před změnou je do sešny uložena hodnota heše původního souboru a načítání nových událostí se spouští pouze v případě, že změna skutečně nastala. Pro vytvoření heše používám hešovací funkci SHA1. Před samotným načítáním nových událostí se nejdříve provede vymazání starých událostí, aby následná analýza neztratila na přehlednosti.

```
1  module Unified2ParseHelper
2    def self.parse_events(user)
3
4      if signatures_file.present? && generators_file.present? &&
           classifications_file.present? && logs_file(user).present?
5        ...
6        @unified2_file = logs_file(user)
7        Unified2.read(@unified2_file) do |event|
8          existing_event = AcidEvent.all_events_by_user_and_timestamp(event.
               event_time)
9          if !existing_event.present?
10           acid_event = user.acid_events.new
11           ...
12           acid_event.source_port = event.source_port
13           acid_event.save!
```

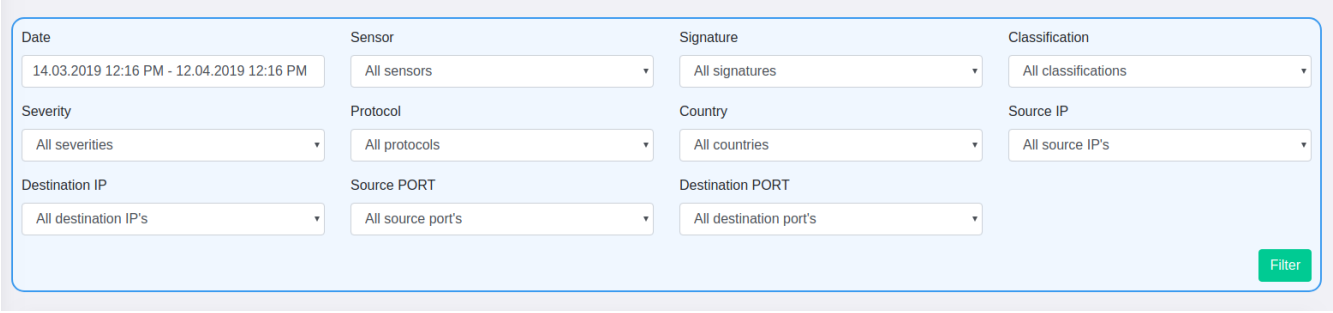
```
14          # ukládání extra informací a paketů pokud jsou k dispozici
15          if event.extras.present?
16            event.extras.each do |extra|
17              acid_event.extras.create!(:name => extra.name, :value => extra.
                    value)
18            end
19          end
20
21          if event.packets.present?
22            event.packets.each do |packet|
23              acid_packet = acid_event.acid_packets.new(:hexdump => packet.
                    hexdump, :hex => packet.hex, ... )
24              if packet.ip_header.present?
25                acid_packet.ip_header = true
26                acid_packet.ip_ver = packet.ip_header[:ip_ver]
27                ...
28                acid_packet.payload = packet.payload
29              end
30              acid_packet.save!
31            end
32        ...
```

Výpis 4: Pseudokód pomocné metody pro ukládání událostí do databáze

6.6.3 Filtrování

Nad načtenými událostmi máme možnost provádět různé filtrace. Filtrace je jinými slovy třídění a vyhledávání v načtených událostech. Kromě samotné filtrace je k dispozici také řazení událostí podle data od nejnovější události k nejstarší a od nejstarší události k nejnovější. Řadit lze i podle tzv. „severity“ což je závažnost dané události. Můžeme si události seřadit od nejzávažnějších, k těm nejméně závažným, nebo naopak. Tyto způsoby řazení fungují i nad vyfiltrovanými událostmi. Filtraci lze provádět na základě data. Datum je ve formátu rozsahu, který si zvolíme, dále podle senzoru, signatury (neboli vzoru), klasifikace události, závažnosti události, protokolu, země vzniku události, zdrojové IP adresy, cílové IP adresy, zdrojového portu a nebo podle portu cílového. Filtrování se provádí jen nad událostmi přihlášeného uživatele.

Pro implementaci filtrů jsem nevyužila žádný pomocný nástroj. Požadavek na filtrování událostí zasílám formou Ajax volání a překresluji pouze část stránky obsahující události, což filtrování výrazně urychluje.



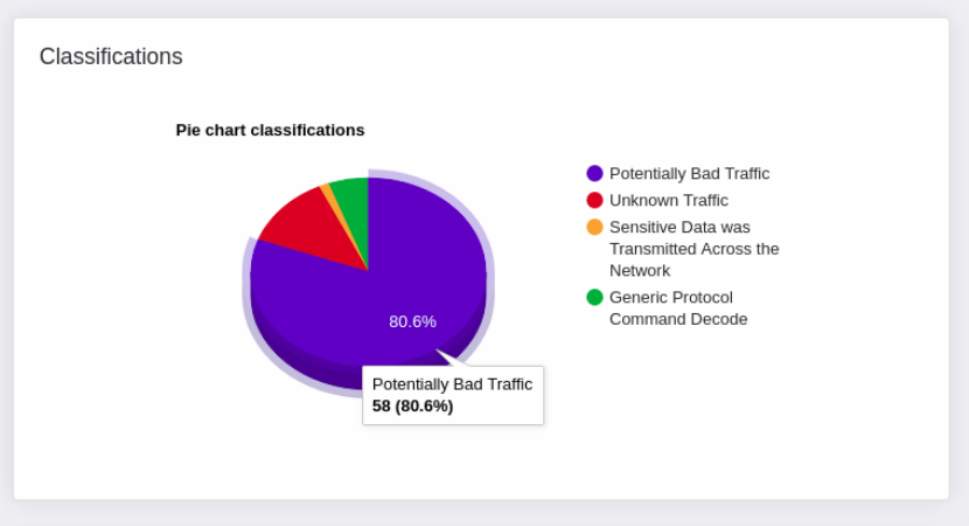
Obrázek 9: Možnosti filtrování

6.6.4 Grafy

Veškeré načtené události u daného uživatele jsou také zpracovávany formou grafů. Pro tuto formu zpracování jsem využila nejhojněji využívané Google grafy. Pro tuto funkcionalitu je ale nutno připojení k internetu.
Po načtení dat vznikne celkem devět grafů. Grafy jsou následující:

1. Graf podílů událostí podle jejich klasifikace.
2. Graf podílů událostí podle signatur.
3. Graf podílů událostí podle závažnosti dané zachycené podezřelé aktivity.
4. Graf podílů událostí podle protokolu.
5. Graf podílů událostí podle země jejich vzniku.
6. Graf podílů událostí podle zdrojové IP adresy.
7. Graf podílů událostí podle cílové IP adresy.
8. Graf podílů událostí podle zdrojových portů.
9. Graf podílů událostí podle cílových portů.

Všechny tyto grafy jsou koláčového typu a můžeme z nich vyčíst poměry sledovaných parametrů k celkovému poměru a vidíme také reálné počty zastoupení těchto sledovaných parametrů. Ukázku takového grafu můžete vidět na obrázku 6.



Obrázek 10: Ukázka grafu: rozdělení událostí podle klasifikace

6.6.5 WhoIs Informace

Aplikace poskytuje informace o IP adrese ze serveru WhoIs. Tyto informace jsou uloženy v databázi aplikace a jsou k dispozici nejen u adresy zdrojové, ale také u adresy cílové. WhoIs nformace jsou dostupné u IP adres verze 4 i 6 a poskytují informace o tom, v jakém bloku se adresa nachází a především pak informace o vlastníkovi dané IP adresy. Pro získání těchto informací byl využit gem WhoIs[8].

6.7 Podrobnější popis obsahu samostatných stránek webového rozhraní

Výsledné rozhraní vytvořené v praktické části a jeho funkce sice byly již popsány výše, ale ne zcela podrobně. Proto jsem se rozhodla věnovat tuto sekci podrobnějšímu popisu samostatných stránek pro lepší orientaci v aplikaci.

6.7.1 Registrace, přihlášení a zapomenuté heslo

Tyto stránky slouží pro vytvoření uživatele, jeho přihlašování a možnost vytvoření nového hesla v případě, že ho uživatel zapomene. Pokud uživatel není přihlášen, tak bude na tuto stránku neustále přesměrováván až do doby, dokud se nepřihlásí, nebo nezaregistruje.

6.7.2 Úprava profilu uživatele

Úprava profilu je samostatná stránka, která vypadá v podstatě stejně jako stránka s registrací, obsahuje ale již vyplněné údaje a umožňuje jejich změnu. Pro změnu údajů je třeba vyplnit stávající heslo. Pro návrat do aplikace slouží tlačítko „Back to the application“.

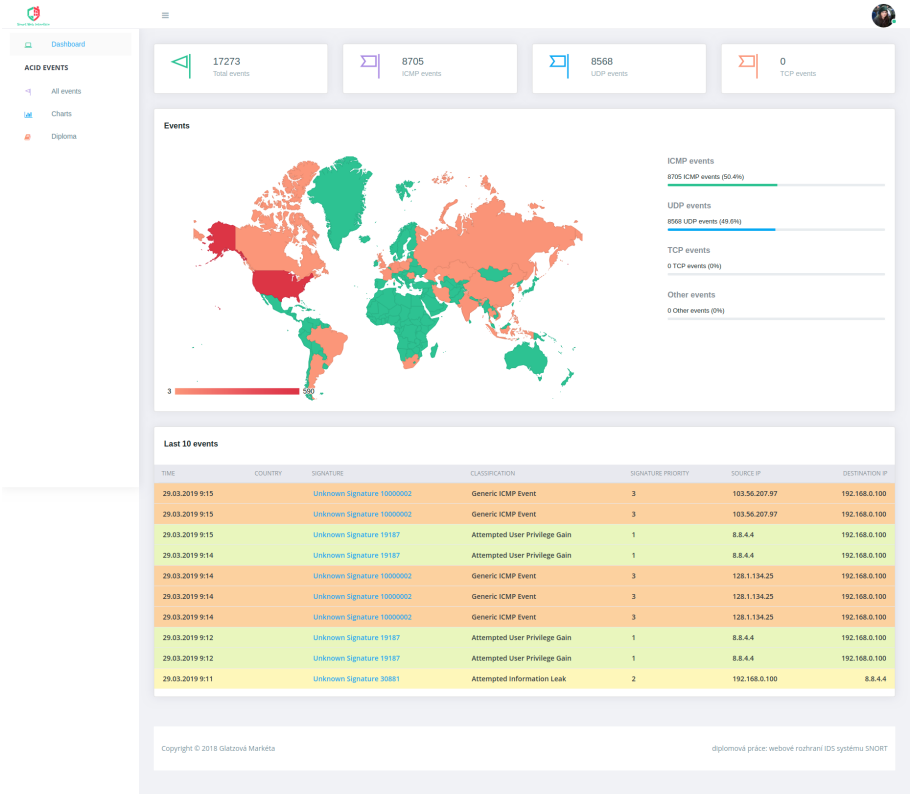
[8] Gem WhoIs, detailní popis naleznete v sekci 6.2.1.4

6.7.3 Nastavení

Dokud uživatel nenastaví aplikaci na stránce s nastavením, tak bude neustále na tuto stránku přesměrováván. Stránka obsahuje dva tabulátory a v každém z těchto tabulátorů je obsažen rozdílný formulář. Uživatel si může vybrat způsob načítání mezi interním (cestou k souboru), nebo externím (vložením souboru) a v závislosti na zvolené možnosti se provádí kontrola správnosti souboru. Po správném nastavení výstupního souboru se již vytváří samotné události.

6.7.4 Hlavní strana

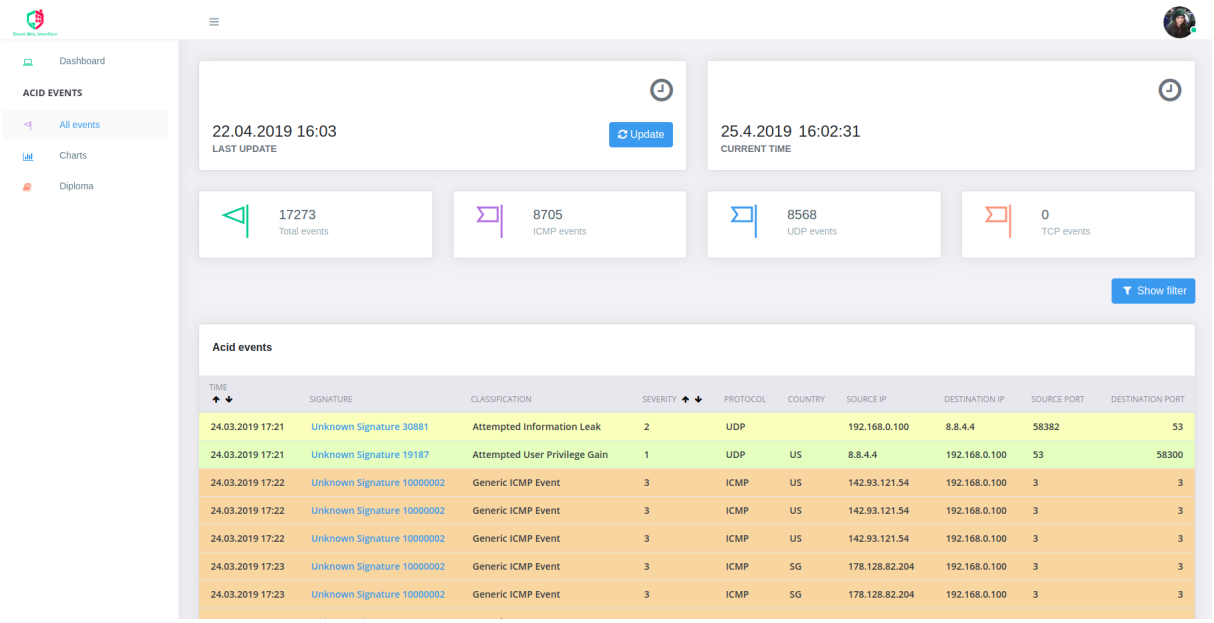
Hlavní strana aplikace je strana na kterou je uživatel přesměrován hned po správném načtení událostí. Nahoře můžeme vidět počet událostí v závislosti na daném protokolu a hned pod těmito informacemi podrobnější informace zobrazeny na mapě a procentuální poměr událostí dle protokolů. Mapa je barevně označena, kde zelená znamená, že z dané země nebyl zachycen žádný škodlivý provoz a červená pak znamená místo odkud byl nějaký takovýto provoz zachycen. Kolik těchto událostí na daném místě bylo ovlivňuje sytost barvy. Barva přechází ze světle červené až po sytě červenou a z mapy lze jednoduše vyčíst také přesný počet daných událostí. Na této stránce je také vyobrazeno posledních deset událostí a to přímo pod touto mapou. Události jsou barevně rozlišeny podle jejich závažnosti a po kliknutí na samotnou událost se dostaneme na stránku s podrobnostmi o útoku.



Obrázek 11: Dashboard

6.7.5 Hlavní strana se všemi událostmi

Hlavní strana se všemi událostmi je asi nejdůležitější stránkou celé aplikace. Nahoře můžeme vidět datum poslední aktualizace s možností provést novou aktualizaci a hned vedle vidíme pak momentální čas pro lepší přehled. Pod těmito informacemi se nachází informace o počtů událostí v závislosti na protokolu, stejně jako na stránce hlavní. Dále zde nalezneme tlačítko „Show filter“, které umožňuje zobrazit, nebo schovat formulář pro vyhledávání. Pod tímto formulářem se již nachází výpis událostí závislém na vybraných filtrech, popřípadě i způsobech řazení.



Obrázek 12: Hlavní strana s událostmi

6.7.6 Zvolená událost

Tato stránka obsahuje kompletní informace o vybrané události a také mapu s označením místa odkud útok přišel. Informace jsou řazeny do tří hlavních sekcí.

Základní sekce obsahuje samotné informace o události, jako například datum a čas zachycení, vzor nebezpečné události, její klasifikace a úroveň závažnosti, protokol, zdrojová a cílová IP adresa a port, adresa původu včetně souřadnic a podobně. V části druhé se nachází veškeré události o paketech včetně jejich obsahu, IP hlavičkách a také extra informací, pokud jsou k dispozici. Poslední část se věnuje informacím ze serveru WhoIs, které jsou dostupné jak pro IP adresu zdrojovou, tak i pro IP adresu cílovou.

6.7.7 Strana obsahující grafické výstupy

Strana obsahující grafické výstupy obsahuje pouze samotné grafy, kterých je celkem devět. Podrobnější výpis jednotlivých grafů a další informace byly zmíněny již výše [9].

6.8 Otestování navrženého řešení

V této části práce jsem se rozhodla provést testy navrženého rozhraní. Jako pozorované parametry jsem si vybrala rychlosti načítání stránek v závislosti na počtu událostí a průměrné doby zpracování vybraných požadavků. Testy byly provedeny na zařízení, které má následující parametry:

- Operační systém Ubuntu 16.04
- Paměť RAM 15,6 GiB
- Procesor Intel® Core™ i5-4210U CPU @ 1.70GHz 4
- Grafická karta Intel® Haswell Mobile

Průměrná doba načítání [ms]		
Sledovaný parametr	při 72 událostech	při 494 událostech
Filtrace událostí	4.39	30.1
Načtení seřazených událostí	4.7	28.5
Načtení hlavní strany aplikace	24.0	45.3
Načtení hlavní strany s událostmi	12.4	34.1
Načtení stránky samotné události	3.0	3.0
Načtení strany obsahující grafy	84.5	152.5

Tabulka 3: Porovnání doby načítání při různých počtech událostí

Průměrná doba zpracování [ms]	
Vytvoření databáze	130.7
Registrace / vytvoření nového uživatele	1.1
Přihlášení / načtení existujícího uživatele	0.9
Zpracování 1 události	8.53
Zpracování souboru se 72 událostmi	613.7
Zpracování souboru se 494 událostmi	4277.4

Tabulka 4: Průměrné doby zpracování jednotlivých požadavků

[9] Grafy, detailní popis naleznete v sekci 6.6.4

7 Srovnání výstupů s dalšími grafickými a webovými rozhraními

Pro srovnání výstupů s rozhraním SnortWebInterface jsem vybrala celkem tři starší rozhraní. Těmito rozhraními jsou SnortAlog, Base a Sguil. Níže popisuji jejich odlišnosti detailněji.

7.1 Platforma

Platformy využívané pro vývoj webových rozhraní jsou různorodé. SnortWebInterface je založen na programovacím jazyce RubyOnRails, SnortAlog je založený na Perlu, Base využívá programovací jazyk PHP a Sguil zase tcl/tk. Z hlediska platformy jsem tedy pro výběr rozhraní určených k porovnání vybrala rozhraní zcela různorodé. Nejprostším rozhraním je SnortAlog, jelikož se jedná o pouhý skript napsaný v Perlu. Tato skutečnost dělá sice ze SnortAlogu rozhraní velice snadné na instalaci, ale jeho možnosti jsou velmi omezené.

7.2 Podpora IPv6

Rozhraní mají běžně problém s událostmi, které mají zdrojovou, nebo cílovou IP adresu verze 6. Nepodařilo se mi najít žádné rozhraní, které by tento problém bylo schopno vyřešit. Tento problém také způsobuje fakt, že Barnyard, který využívá většina rozhraní pro převod dat do databáze, nepodporuje IPv6 adresy. SnortWebInterface na rozdíl od ostatních rozhraní využívá svou vlastní databázi a je schopen pracovat i s IP adresami verze 6. Pro převod adres jsem použila gem Ipaddress[10]

7.3 Informace o událostech

Základní informace o událostech jsou pro všechny rozhraní stejné. Těmito základními informacemi jsou senzor, klasifikace, podpis útoku, závažnost útoku, čas zachycení, protokol (TCP, UDP, ICMP a ostatní), zdrojová a cílová IP adresa, zdrojový a cílový port. Kromě těchto informací nabízí rozhraní SnortWebInterface, SnortAlog, Base i Sguil rozšířené informace o IP adresách ze serveru WhoIs. Jelikož ale SnortWebInterface jako jediný podporuje i IPv6, tak i jako jediný nabízí také WhoIs informace o adresách této verze. SnortWebInterface si kromě samotných WhoIs informací do databáze také ukládá souřadnice původu události, které následně využívá k přesnému určení místa na Google mapě[11]. Dále si navíc ukládá kompletní adresu zjištěnou ze zdrojové IP adresy a kód země. Zdrojový kód země využívá SnortWebInterface pro zobrazení grafu ve formě mapy na hlavní stránce aplikace. Z této mapy je patrné odkud pochází nejvíce nebezpečného provozu. Sguil si kromě základních informací ukládá ještě svůj status, který přiřazuje ke konkrétním událostem. Nezařazené události mají prioritu číslo 0. Base a SnortAlog nabízí pouze základní informace o událostech.

[10] Gem Ipaddress, detailní popis naleznete v sekci 6.2.1.2
[11] Google mapa, detailní popis naleznete v sekci 7.5

7.4 Informace o paketech

Z vybraných rozhraní není schopno zobrazovat detailnější informace o paketech pouze jediné rozhraní a tímto rozhraním je SnortAlog. SnortAlog dokáže zobrazit pouze základní informace z upozornění jako například zdrojovou a cílovou IP adresu, zdrojový a cílový port, klasifikaci útoků, závažnost útoku a pod. SnortWebInterface je schopen poskytnout úplné informace o paketech včetně informací z IP hlavičky paketu. Informace z IP hlavičky je dále schopen zobrazit pouze Sguil. Sguil v této stránce vyčnívá nad ostatními rozhraními, jelikož je schopen nejen zobrazit úplné informace o zachycených paketech včetně jejich obsahu podobně jako SnortWebInterface, ale je také schopen zobrazit dané pravidlo Snortu, které bylo použito pro zachycení události. Oproti tomu Base dokáže pracovat pouze s datovou části paketu, ale informace o IP hlavičce jsem nenalezla.

7.5 Vizualizace výstupů

Většina rozhraní poskytuje uživateli přehledné analýzy ve formě grafů, Sguil však toto neumožňuje. Je veliká škoda, že má Sguil tento nedostatek, protože je to jinak dle mého názoru velmi příjemný nástroj pro analýzu událostí. SnortAlog používá pro vizualizaci kombinaci ne příliš vzhledných grafů a tabulek sloužících k jejich doplnění. Tento způsob vizualizace mi nepřipadá nejvhodnější a působí na mne velice zastaralým dojmem. Oproti tomu Base nabízí vcelku kvalitní způsob vizualizace s možností nastavení parametrů u výstupního grafu, čímž se stává jedinečným. SnortWebInterface sice neumožňuje uživateli nijak upravovat grafické výstupy, ale nabízí oproti ostatním moderní grafy od společnosti Google, které jsou mezi vývojáři stále oblíbenějšími. Jako jediný také ve výpisu u vybrané události nabízí Google mapu s vyznačeným místem původu události. Mapa funguje dynamicky, uživatel si tedy může přepnout mapu do satelitního režimu, přiblížit a oddálit, posunout a podobně.

7.6 Vzhledová stránka rozhraní a přehlednost

Pokud se zaměřím na vzhledovou stránku vybraných rozhraní, tak si troufám říct, že SnortWebInterface je nejmodernějším a nejpřehlednějším ze všech vybraných rozhraní. Jako jediný využívá moderní technologie pro tvorbu designu a odpovídá také moderním standardům 21. století. Nejméně přehledným rozhraním s nejvíce zastaralým vzhledem je dle mého názoru SnortAlog. Výstup SnortAlogu obsahuje množství tabulek, které mi přijdou nic nevypovídající a nepřehledné. Base a Sguil jsou na tom o poznání lépe, ale mohou působit na běžného uživatele příliš složitě a příliš „programátorským“ vzhledem. Base obsahuje dle mého názoru příliš mnoho samostatných stran a hledání požadovaných parametrů ve výstupech může být tedy ze začátku používání zdlouhavé. Častějším používáním se sice uživatel již naučí, kde hledat dané informace, ale myslím si, že to některé uživatele může odradit již na začátku. Nejvíce se u Base nepovedlo vyhledávání, které je opravdu nepřehledné a příliš složité. Například pokud chceme vyhledat události pouze s určitou signaturou, tak nelze jen vybrat požadovanou signaturu z možností a

události vyhledat, ale je třeba znát kompletní název této signatury. Toto ale bohužel není vše, Base používá také pro porovnání operátory, kterým nemusí každý rozumět. Například ne každý uživatel ví, že „!=“ je operátor značící nerovnost. Sguil sice tyto operátory nevyužívá, ale pořád neposkytuje zcela jednoduché vyhledávání v událostech jako tomu je u SnortWebInterface. Sguil je přesto výborným nástrojem, jelikož obsahuje oproti ostatním rozhraním poskytuje některé informace navíc, není ale podle mne vhodný pro nezkušené uživatele. Vzhledem nápadně připomíná paketový analyzátor Wireshark, což ocení především síťaři, běžný uživatel však nikoliv. SnortWebInterface jsem se snažila oproti ostatním rozhraním udělat tak, aby byl dostatečně přehledný i pro méně zkušené uživatele a zároveň poskytoval veškeré důležité informace na jednom místě a myslím si, že se mi to povedlo.

	Platforma	Podpora IPv6	Informace o paketech	Výstupy v podobě grafů	Možnosti filtrování	Informace ze serveru WhoIs	Zobrazení útoku na mapě	Informace z hlavičky IP
SnortWebInterface	Ruby on Rails	✓	✓	✓	✓	✓	✓	✓
SnortAlog	Perl			✓		✓		
Base	PHP		✓	✓	✓	✓		
Sguil	tcl/tk		✓		✓	✓		✓

Tabulka 5: Shrnutí vybraných grafických rozhraní pro Snort

8 Závěr

Cílem této diplomové práce bylo vytvořit nástavbu aplikace Snort umožňující přehlednou analýzu logů skrz webové rozhraní. V první části práce byly nejdříve objasněny základní pojmy týkající se IDS systémů, jejich rozdělení podle umístění v síti, způsoby jejich detekce a rozdíly oproti systémům IPS. Dále následoval podrobnější popis a porovnání vybraných IDS systémů. Samotnému IDS systému Snort byla věnována celá kapitola, která popisuje režimy a komponenty IDS systému Snortu, jeho pravidla a detailně se věnuje jeho výstupním modulům. Ve druhé části byly představeny další vybrané systémy umožňující vizualizaci protokolů systému Snort, které zároveň sloužily jako inspirace pro výsledné rozhraní. Třetí část byla věnována již samotnému výslednému rozhraní. Pro tvorbu rozhraní byl použit programovací jazyk Ruby On Rails. Ruby on Rails nabízí mnoho volně dostupných doplňků ve formě gemů, což jsem při tvorbě práce využila. Veškeré důležité gemy byly popsány podrobněji. Výsledné webové rozhraní bylo navrženo tak, aby byla jeho instalace co nejsnadnější a aby bylo schopno fungovat samostatně a nepotřebovalo další software pro převod dat do databáze, jakož tomu je u většiny ostatních rozhraní. Výsledné rozhraní má tedy vlastní databázi, která je v práci detailně popsána včetně vyobrazení databázového diagramu. Pro maximální zjednodušení instalace byl využit Docker, který funguje jako virtuální stroj a využívá tzv. kontejnéry, ve kterých běží potřebné služby. Jedinými prerekvizitami pro chod aplikace tedy jsou instalace Docker Compose a připojení k internetu (pro komunikaci s Google grafy a mapami). Cílem bylo vytvořit rozhraní, které bude působit moderním dojmem, z toho důvodu byly také při tvorbě rozhraní využity moderní technologie pro tvorbu vizuální podoby. Inspirovala jsem se také moderními šablonami určenými pro administrátorské stránky, výsledný vzhled je ale originální. Ke konci třetí části byly blíže popsány funkce samotné aplikace jako například způsob načítání dat do aplikace a byly provedeny testy rychlosti načítání dat. Poslední část byla věnována srovnání výstupů s dalšími třemi systémy umožňující vizualizaci protokolů systému Snort. Oproti těmto třem rozhraním skončil SnortWebInterface velice dobře. Jako jediný například nabízí podporu IP adres verze 6, jako jediný dále nabízí zobrazení původu události na dynamické mapě. Jako jeden z mála dokáže zobrazit také informace z IP hlavičky paketu a z hlediska vzhledové stránky ho lze nazvat nejmodernějším.

Výsledné rozhraní je ve verzi 1.0 a je plánováno jeho další rozšiřování. Jedná se například o úpravu aplikace tak, aby si byla sama schopna sloučit výstupní soubory a vybrat pro načtení dat sloučený soubor. Aplikace pak bude fungovat samostatněji a uživateli odpadne další krok, který je momentálně potřeba provést. Jako jedna z možných budoucích funkcí by mohla také například být instalace Snortu přímo v Dockeru. Pokud by Snort běžel přímo v Dockeru v rámci aplikace, tak by to nejen výrazně ulehčilo uživateli práci s jeho instalací, ale byla by také možnost užšího propojení se Snortem. Naskýtala by se například možnost upravovat a vytvářet pravidla přímo v aplikaci. Funkce úprav tykajících se přímo Snortu by byly omezeny pouze pro uživatele s administrátorskými právy. Tato role je již v aplikaci připravena. Jelikož existují také zajímavé

paketové analyzátory fungující v reálném čase a jsou napsány v jazyce Ruby On Rails, tak by do budoucna bylo možné například samotný Snort rozšířit také o jeden z těchto analyzátorů. Tuto analýzu provozu v reálném čase by mohli spouštět pouze uživatelé s administrátorskou rolí. Pokud by se aplikace začala rozšiřovat, tak by bylo vhodné naimplementovat chat mezi uživateli, aby mezi sebou mohli komunikovat. Praktické by bylo také přidání TODO listu, do kterého by si mohli uživatelé přidávat úkoly, které by například byly spojeny s objevenými hrozbami v síti a kroky k dosažení lepšího zabezpečení sítě. Z hlediska stránky vzhledové bych chtěla do nastavení aplikace přidat uživateli možnost nastavit si barvu pro označení událostí v závislosti na jejich závažnostech. Možností dalšího vývoje existuje ale mnohem více a z vytvořeného rozhraní lze v budoucnosti vytvořit velmi rozsáhlý informační systém.

Literatura

[1] Chris McNab: *Network Security Assessment: Know Your Network*, O'Reilly Media, 3 edice, 2016, ISBN 978-1491910955

[2] Rafeeq Rehman: *Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID*, Prentice Hall Professional, 2003, ISBN 978-3659982011

[3] Dr Anand Nayyar: *The Best Open Source Network Intrusion Detection Tools [online]*, Dostupné z: https://opensourceforu.com/2017/04/best-open-source-network-intrusion-detection-tools

[4] Renaud Larue-Langlois: *Top 10 Intrusion Detection Tools: Your Best Free Options for 2019 [online]*, Dostupné z: https://www.addictivetips.com/net-admin/intrusion-detection-tools/

[5] *Oficiální dokumentace pro Surikatu [online]*, Dostupné z: https://suricata-ids.org/docs/

[6] *Oficiální uživatelská příručka Surikaty [online]*, Dostupné z: https://suricata.readthedocs.io

[7] *Oficiální dokumentace pro Snort [online]*, Dostupné z: https://www.snort.org/documents/

[8] *Oficiální uživatelská příručka Snortu [online]*, Dostupné z: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/

[9] Rapid7: *Understanding and Configuring Snort Rules [online]*, Dostupné z: https://blog.rapid7.com/2016/12/09/understanding-and-configuring-snort-rules/

[10] *Oficiální dokumentace pro OSSEC [online]*, Dostupné z: https://www.ossec.net/docs/

[11] *Blog věnující se informacím o IDS Zeek [online]*, Dostupné z: https://blog.zeek.org/

[12] *Oficiální uživatelská příručka Security Onion [online]*, Dostupné z: https://securityonion.readthedocs.io

[13] *Oficiální stránky IDS OpenWIPS-ng [online]*, Dostupné z: http://openwips-ng.org/index.html

[14] Oficiální Blog věnující se Snortu: *Rozhraní pro Snort [online]*, Dostupné z: https://blog.snort.org/2011/01/guis-for-snort.html

[15] Jérémy Chartier: *Stránky věnující se rozhraní SnortAlog [online]*, Dostupné z: http://jeremy.chartier.free.fr/snortalog

[16] *Oficiální stránky AlienVault věnující se produktu OSSIM [online]*, Dostupné z: https://www.alienvault.com/products/ossim

[17] David Bianco: *Nejčastější dotazy týkající se rozhraní Sguil [online]*, Dostupné z: http://nsmwiki.org/Sguil_FAQ

[18] *Obrázek rozdílu zapojení IDS a IPS [online]*, Dostupné z: https://www.digitalplanet.ie/ips-ids/

Seznam příloh

- snort_web_interface – Složka se samotným výsledným grafickým rozhraním pro Snort.
- obrázky – Složka obsahující veškeré obrázky, které byly použity při tvorbě této diplomové práce.
- testovací data – Složka se souborem typu Unified2 vhodnému k testování aplikace.